Verification of a Viscous Computational Aeroacoustics Code using External Verification Analysis

Daniel Ingraham*
Ray Hixon†
University of Toledo, Toledo, OH 43606

The External Verification Analysis approach to code verification is extended to solve the three-dimensional Navier-Stokes equations with constant properties, and is used to verify a high-order computational aeroacoustics (CAA) code. After a brief review of the relevant literature, the details of the EVA approach are presented and compared to the similar Method of Manufactured Solutions (MMS). Pseudocode representations of EVA's algorithms are included, along with the recurrence relations needed to construct the EVA solution. The code verification results show that EVA was able to convincingly verify a high-order, viscous CAA code without the addition of MMS-style source terms, or any other modifications to the code.

Nomenclature

x, y, z	Cartesian coordinates
t	Time
$u, \hat{u}$	Exact and manufactured solution to the linear advection equation
f, g	Arbitrary differentiable functions
a	Advection speed for the linear advection equation
b	Arbitrary constant used in $\hat{u}$
S	Linear advection manufactured solution source term
ρ	Density
σ	$1/\rho$
u	x-velocity
v	y-velocity
w	z-velocity
p	Pressure
Φ, Θ	Navier-Stokes mechanical and conductive viscous terms, respectively
X	x, y, or z
U	σ, u, v, w, or p
γ	Ratio of specific heats
R	Ideal gas constant
k	Thermal conductivity
μ	Dynamic viscosity
ϵ	Error norm
Φ, ϕ	Reference and PDE code solutions, respectively
h	Discretization measure
A, B	h-independent coefficients
p	Observed order-of-accuracy
ϕ	Primitive flow variable initial condition (σ, u, v, w, p)

*Former Graduate Research Assistant, Mechanical, Industrial, and Manufacturing Engineering (MIME) Department, Student Member AIAA.
†Associate Professor, Mechanical, Industrial, and Manufacturing Engineering (MIME) Department, Senior Member AIAA.

$\overline{\phi}$, $\widetilde{\phi}$	Constants in ϕ
x_0, y_0, z_0	Constants in ϕ
k_x, k_y, k_z, θ	constants in ϕ
k	Wavenumber of a single Fourier component
$(k\Delta x)^*$	Numerical wavenumber of a spatial differencing scheme
ϵ	Spatial differencing scheme's relative error
ω	Frequency of a single Fourier component
N	Number of grid points in test case grid
G	Time marching scheme amplification factor
θ, ϑ	Local and global time-marching error, respectively

I. Introduction

As computational aeroacoustics codes become more sophisticated and widespread, the need for quantifying their accuracy becomes more important. The field of Verification & Validation (V&V, cf. the recent book[1] on the subject) is focused on developing techniques for assessing the correctness and reliability of scientific codes, including determining the accuracy of their numerical algorithms (i.e., verification), and the closeness with which their output correspond to nature or experiment (i.e., validation).

Code verification, the crucial first step of V&V, is an investigation into whether a CAA code (or any PDE solver) is "solving the equations right.[2]" Specifically, a *code* can be deemed verified if it is rigorously shown to solve its intended PDE to the formal order-of-accuracy of its numerical schemes. Code verification is distinct from *solution* verification, which is the process of estimating the amount of numerical error in a PDE code's solution. See[1,3] and especially[4] for clear discussions of code verification, solution verification, and other V&V concepts.

Code verification requires evaluating the error in a PDE code's solution, which in turn requires the comparison of a PDE code's output to a reference solution. One approach to obtaining such a reference solution is called the method of exact solutions (MES, a term likely coined by Knupp and Salari[5]). One simply finds an exact solution to the PDE the code in question solves somewhere in the literature, applies the PDE code to the problem to which the exact corresponds, and compares the code's output to the exact solution. For example, to verify a PDE code that solves the linear advection equation,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{1}$$

the exact solution

$$u(x,t) = f(x - at), \tag{2}$$

where f is any differentiable function, is taken as the reference solution and used to calculate the PDE code's error.

Equation (2) is a quite general solution to (1): one has considerable flexibility in choosing f, and thus the form of the solution. Unfortunately, known exact solutions to non-trivial PDEs are usually not as flexible, instead tending to be either too simple to adequately exercise all the capabilities of the code, or are tedious to evaluate numerically. An example of an exact solution that is simultaneously too simple and too complex is provided by Knupp and Salari,[5] who reproduce the exact solution to unsteady homogeneous heat conduction in a sphere. Evaluating the solution involves two infinite series, a triple integral, and various flavors of Bessel functions, Bessel function roots, and Legendre polynomials. Such a solution, however, would be inadequate for verifying a general heat conduction code that can handle arbitrary geometries and non-constant physical properties, since it (the solution) does not exhibit any of these characteristics.

The Method of Manufactured Solutions (MMS) was developed to overcome this limitation of MES. First introduced by Steinburg and Roache,[6] MMS can be used to make nearly any expression (within reason) an exact solution to a code's PDE at the cost of introducing an additional source term to PDE, and thus the code itself. Excellent explanations of the MMS technique are available in the literature,[1,5,7–9] but a brief example of its use is given here. Suppose, again, that a PDE code solves the linear advection equation (1), and one wishes to use MMS to verify the code. The reference ("manufactured") solution is specified next — or, as Roache[9] points out, *before* even looking at the PDE. A simple choice is made

$$\hat{u}(x,t) = \sin(x - bt) \tag{3}$$

where $\hat{u}$ is the reference solution and b an arbitrary constant. The proposed solution is then substituted into the governing equation to find the MMS source term:

$$\begin{aligned} S(x,t) &= [\sin(x-bt)]_t + a\,[\sin(x-bt)]_x \\ &= -b\cos(x-bt) + a\cos(x-bt) \\ &= (a-b)\cos(x-bt). \end{aligned} \tag{4}$$

The source term $S(x,t)$ is finally added to the original governing equation:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = S(x,t) = (a-b)\cos(x-bt). \tag{5}$$

The manufactured solution $\hat{u} = \sin(x-bt)$ is an *exact* solution of the modified equation (5), and can be used as a reference solution when verifying any PDE code that can be made to solve (5). (Of course, if one chose $b = a$, then $S(x,t) = 0$ and $\hat{u}$ is an exact solution to the original PDE. Doing the same for more complex PDEs is generally not possible.)

The Method of Manufactured Solutions has enjoyed fairly wide adoption in the CFD community by those concerned with V&V, having been used to verify Finite Volume method CFD codes[10–13] solving both the Euler and Navier-Stokes equations on unstructured meshes, fluid-structure interaction (FSI) codes,[14–16] and RANS codes with turbulence models.[17,18] Instances of its use in non-CFD applications are less prevalent, but do exist. Merroun et al.[19] showed how MMS could be used with a code simulating convective heat transfer in multiple channels (the code's application was nuclear reactor cooling), and Pautz[20] communicated results of applying MMS to a finite element code solving equations with applications to nuclear transport. Also, one of the FSI examples previously mentioned[16] had its origin in biomechanics problems.

Despite its generality and many proponents in the V&V community, MMS appears to be used to verify new and existing PDE codes infrequently. For example, a review of recent issues of the *Journal of Computational Physics* (JCP, issue 274) and *Computers & Fluids* (C+F, issue 103) was performed, and the code verification technique(s) used by the authors, if any, were noted. Results are summarized in Figure 1, where the number of papers using MES and MMS are displayed, along with Code Comparison,[21] a less-rigorous code verification technique that involves comparing the output of one code with another. Clearly, the Method of Manifactured Solutions is the least-popular code verification technique. The reason for this is likely related to the MMS source term that must be added to the PDE code, which does tend to be complex for non-trivial governing equations (though computer-algebra systems like Mathematica, Matlab make this complexity much more manageable). The ideal code verification tool would allow the user to create a reference solution consisting of elementary, easy-to-evaluate functions, but would not require any changes to the equations the PDE code solves — the flexibility of MMS with the unobtrusiveness of MES. External Verification Analysis[22] is the authors' attempt at such a tool.

II. External Verification Analysis

External Verification Analysis' (EVA) point of departure is related to, but different from, the Method of Manufactured Solutions. Instead of specifying the *complete* solution to a PDE up front (i.e., the MMS approach), a solution on a reduced-dimensional surface is chosen, and then expanded beyond the initial surface using a Taylor series. The coefficients of the Taylor series are calculated using the PDE itself and Cauchy-Kowalewski recursion.[23,24] For example, with the same PDE from the MMS demonstration, the solution on the $t = 0$ "surface" is specified,

$$u(x,0) = u|_0(x). \tag{6}$$

If a solution similar to that of the MMS demonstration is desired, $u|_0(x) = \sin(x)$. The solution for $t \neq 0$ is found using the Taylor series

$$u(x,t) = u|_0 + t\left.\frac{\partial u}{\partial t}\right|_0 + \frac{t^2}{2}\left.\frac{\partial^2 u}{\partial t^2}\right|_0 + \frac{t^3}{6}\left.\frac{\partial^3 u}{\partial t^3}\right|_0 + \cdots \tag{7}$$

Expressions for the temporal derivatives in the coefficients of (7) are found by performing Cauchy-Kowalewski (CK) recursion on the linear advection PDE. First, the temporal derivative is isolated:

$$\frac{\partial u}{\partial t} = -a\frac{\partial u}{\partial x}, \tag{8}$$

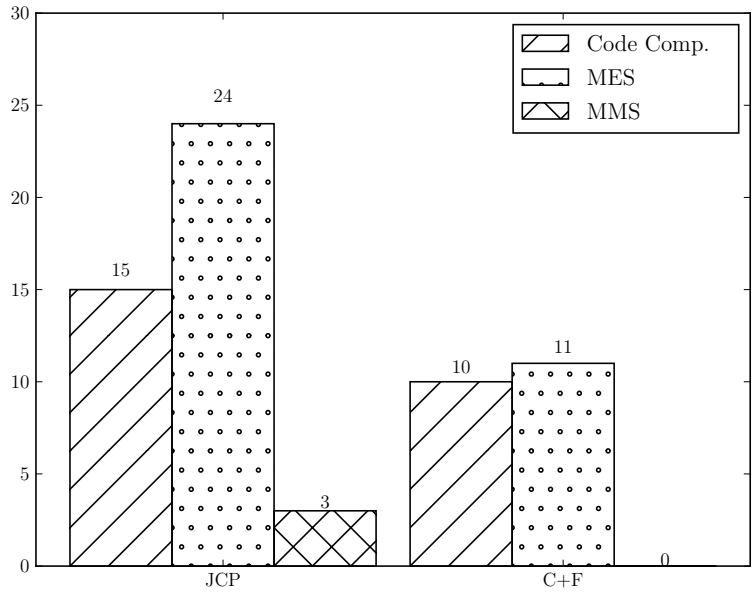

Figure 1. Counts of code verification techniques used in recent issues of the *Journal of Computational Physics* (JCP) and *Computers & Fluids* (C+F).

which, after substituting $t = 0$, provides an expression for the first temporal derivative in the Taylor series, i.e.,

$$\left.\frac{\partial u}{\partial t}\right|_0 = -a \left.\frac{\partial u}{\partial x}\right|_0 \tag{9}$$

where $\left.\frac{\partial u}{\partial x}\right|_0$ is found by differentiating (6). The second-order temporal derivative is found by first differentiating (8) with respect to t:

$$\frac{\partial^2 u}{\partial t^2} = -a \frac{\partial^2 u}{\partial x \, \partial t}. \tag{10}$$

An expression for the mixed spatial-temporal derivative $\frac{\partial^2 u}{\partial x \, \partial t}$ in (10) is found again from (8), this time by differentiating with respect to x, i.e.:

$$\frac{\partial^2 u}{\partial x \, \partial t} = -a \frac{\partial^2 u}{\partial x^2}. \tag{11}$$

Taken together, (10) and (11) provide the necessary information to calculate $\frac{\partial^2 u}{\partial t^2}$ at $t = 0$: first $\frac{\partial^2 u}{\partial x^2}$ is found by differentiating (6) twice, which is used to evaluate $\frac{\partial^2 u}{\partial x \, \partial t}$ from (11), which in turn allows one to find $\frac{\partial^2 u}{\partial t^2}$ from (10). This recursive process of successively finding expressions for temporal and mixed spatial-temporal derivatives in terms of pure spatial derivatives can be repeated indefinitely, allowing the Taylor series of (7) to be extended to an arbitrary order.

The preceding discussion shows that there are essentially two phases to CK recursion: first, an expression for a needed derivative of u is found by differentiating the governing equation; second, the temporal order of the terms in the new derivative expression are reduced, again by differentiating the governing equation. This process is described in pseudocode as the recursive procedure CK in Algorithm 1. The procedure takes two arguments: deriv_id, some object that represents the order and dependent variable that should be found (i.e., the programming equivalent of a symbol like $\frac{\partial^5 u}{\partial x^2 \, \partial t^3}$), and deriv_cache, an object that stores the derivatives that have already been found and their numerical values. The routine begins by checking if the value of the derivative deriv_id has been calculated previously and is thus stored inside deriv_cache. If it has, the routine does nothing (**pass**), the rest of the **if** block is skipped, and the program exits the routine.

Algorithm 1 Pseudocode for Cauchy-Kowalewski recursion

```
procedure CK(deriv_id, deriv_cache)
    if deriv_id in deriv_cache then
        pass
    else if GET_TORDER(deriv_id) == 0 then
        deriv_val ← EVAL_IC(deriv_id, deriv_cache)
        STORE_VAL(deriv_id, deriv_val, deriv_cache)
    else
        expr ← GET_EXPR(deriv_id)
        for all local_deriv_id in expr do
            CK(local_deriv_id, deriv_cache)                    ▷ Recursion!
        end for
        deriv_val ← EVAL_EXPR(expr, deriv_cache)
        STORE_VAL(deriv_id, deriv_val, deriv_cache)
    end if
end procedure
```

If deriv_id does need to be found, the CK routine determines if the needed derivative is a pure-spatial derivative (e.g., $\frac{\partial^3 u}{\partial x^3}$) or a mixed spatial-temporal or pure-temporal derivative (e.g., $\frac{\partial^2 u}{\partial x\,\partial t}$ or $\frac{\partial^4 u}{\partial t^4}$), represented in the pseudocode as the check in the GET_TORDER(deriv_id) == 0 part of the **else if** statement. GET_TORDER is imagined to be a function that takes the deriv_id object and returns the order of its temporal derivative (so GET_TORDER($\frac{\partial^3 u}{\partial x^2\,\partial t}$) is 1 and GET_TORDER($\frac{\partial^4 u}{\partial x\,\partial t^3}$) is 3). If deriv_id is indeed a pure-spatial derivative, its numerical value is calculated by the EVAL_IC function and then saved in the deriv_cache object by the STORE_VAR routine, and the routine returns.

The last branch of the **if**/**else if**/**else** block handles the case when deriv_id is a mixed spatial-temporal or pure-temporal derivative than has not been calculated previously. First, an expression for the desired derivative is found by differentiating the governing equation and placed in the expr variable, represented in Algorithm 1 by the line containing the GET_EXPR function, which is the pseudocode equivalent of obtaining (10) or (11) through the differentiation of (8). Next, each term in expr is passed to the CK routine, "winding up" the recursion. The recursive CK call has the effect of finding any unknown derivatives in expr and storing them in deriv_cache. Finally, the "unwinding" of the CK procedure is accomplished by the last two lines of the **else** branch, which find the numerical value of the deriv_id derivative using the previously-calculated derivative data in deriv_cache and the EVAL_EXPR routine, and then save the value in the deriv_cache variable, again using the STORE_VAL routine.

Algorithm 1 could be used to easily implement Cauchy-Kowalewski recursion as part of an EVA tool, especially if a computer algebra system (CAS) is available. Indeed, such a program has been devolved using the Python programming language[25] and SymPy,[26] an open-source Python CAS library, and subsequently used to very the EVA tool itself. While the flexibility provided by a high-level language like Python and CAS library like SymPy is attractive, such flexibility comes at the cost of efficiency. Also, unless a compiler optimizes it, recursive routines tend to be considerably slower than an explicit loop-based representation. For these reasons, unwinding the recursion in Algorithm 1 and finding an expression for the arbitrary-order derivative of the targeted governing equations will be discussed in the next section.

III. EVA with the Navier-Stokes Equations

This work proposes to use the technique presented in the previous section to construct a reference solution to the Navier-Stokes equations with constant properties (i.e., ratio of specific heats, gas constant, viscosity, and conductivity) that can then be used to verify a high-order viscous CAA code. The three-dimensional form of the Navier-Stokes equations with primitive variables $\sigma = 1/\rho$, u, v, w, and p are used with the EVA

tool:

$$\frac{\partial \sigma}{\partial t} + u\frac{\partial \sigma}{\partial x} + v\frac{\partial \sigma}{\partial y} + w\frac{\partial \sigma}{\partial z} - \sigma\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) = 0 \tag{12a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \sigma\frac{\partial p}{\partial x} = \mu\sigma\left(\frac{4}{3}\frac{\partial^2 u}{\partial x^2} + \frac{1}{3}\frac{\partial^2 v}{\partial x\,\partial y} + \frac{1}{3}\frac{\partial^2 w}{\partial x\,\partial z} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{12b}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \sigma\frac{\partial p}{\partial y} = \mu\sigma\left(\frac{4}{3}\frac{\partial^2 v}{\partial y^2} + \frac{1}{3}\frac{\partial^2 u}{\partial x\,\partial y} + \frac{1}{3}\frac{\partial^2 w}{\partial y\,\partial z} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2}\right) \tag{12c}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} + \sigma\frac{\partial p}{\partial z} = \mu\sigma\left(\frac{4}{3}\frac{\partial^2 w}{\partial z^2} + \frac{1}{3}\frac{\partial^2 u}{\partial x\,\partial z} + \frac{1}{3}\frac{\partial^2 v}{\partial y\,\partial z} + \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) \tag{12d}$$

$$\frac{\partial p}{\partial t} + u\frac{\partial p}{\partial x} + v\frac{\partial p}{\partial y} + w\frac{\partial p}{\partial z} + \gamma p\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) = (\gamma - 1)\left(\Theta + \Phi\right). \tag{12e}$$

where

$$\Theta = \frac{\mathrm{k}}{\mathrm{R}}\left(\frac{\partial^2 \sigma}{\partial x^2}p + 2\frac{\partial \sigma}{\partial x}\frac{\partial p}{\partial x} + \sigma\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 \sigma}{\partial y^2}p + 2\frac{\partial \sigma}{\partial y}\frac{\partial p}{\partial y} + \sigma\frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 \sigma}{\partial z^2}p + 2\frac{\partial \sigma}{\partial z}\frac{\partial p}{\partial z} + \sigma\frac{\partial^2 p}{\partial z^2}\right) \tag{13}$$

and

$$\begin{aligned}\Phi = \mu\Bigg(\frac{4}{3}\Bigg[&\frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\frac{\partial w}{\partial z} - \frac{\partial v}{\partial y}\frac{\partial w}{\partial z}\Bigg] \\ &+ \frac{\partial v}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial y} + \frac{\partial w}{\partial y}\frac{\partial w}{\partial y} + \frac{\partial u}{\partial z}\frac{\partial u}{\partial z} + \frac{\partial v}{\partial z}\frac{\partial v}{\partial z} \\ &+ 2\left[\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial w}{\partial y}\frac{\partial v}{\partial z} + \frac{\partial u}{\partial z}\frac{\partial w}{\partial x}\right]\Bigg).\end{aligned} \tag{14}$$

Again, (12)–(14) assume the dynamic viscosity μ, thermal conductivity k, and gas constant R are all constant.

As discussed in the previous section, an explicit expression for the arbitrary-order derivative of the governing equations will allow for a more efficient implementation of the EVA technique. Because each equation in (12) only contains sums of products, all that is needed is nested applications of the Leibniz rule

$$\frac{d^n\left[f(x)\cdot g(x)\right]}{dx^n} = \sum_{m=0}^{n}\binom{n}{m}\frac{d^{n-m}f}{dx^{n-m}}\frac{d^m g}{dx^m}. \tag{15}$$

Applying (15) to the first equation in (12a) (the continuity equation) gives

$$\frac{\partial \sigma}{\partial t} = -\left(u\frac{\partial \sigma}{\partial x} + v\frac{\partial \sigma}{\partial y} + w\frac{\partial \sigma}{\partial z} - \sigma\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right]\right) \tag{16a}$$

$$\begin{aligned}\frac{\partial^{n+1}\sigma}{\partial t^{n+1}} = -\sum_{m=0}^{n}\binom{n}{m}\Bigg[&\frac{\partial^{n-m}u}{\partial t^{n-m}}\frac{\partial^{1+m}\sigma}{\partial x\,\partial t^m} + \frac{\partial^{n-m}v}{\partial t^{n-m}}\frac{\partial^{1+m}\sigma}{\partial y\,\partial t^m} + \frac{\partial^{n-m}w}{\partial t^{n-m}}\frac{\partial^{1+m}\sigma}{\partial z\,\partial t^m} \\ &- \frac{\partial^{n-m}\sigma}{\partial t^{n-m}}\left(\frac{\partial^{1+m}u}{\partial x\,\partial t^m} + \frac{\partial^{1+m}v}{\partial y\,\partial t^m} + \frac{\partial^{1+m}w}{\partial z\,\partial t^m}\right)\Bigg]\end{aligned} \tag{16b}$$

$$\begin{aligned}\frac{\partial^{b+d+f+n+1}\sigma}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^{n+1}} = -\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[&\\ \frac{\partial^{b-a+d-c+f-e+n-m}u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}&\frac{\partial^{a+1+c+e+m}\sigma}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\ + \frac{\partial^{b-a+d-c+f-e+n-m}v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}&\frac{\partial^{a+c+1+e+m}\sigma}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\ + \frac{\partial^{b-a+d-c+f-e+n-m}w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}&\frac{\partial^{a+c+e+1+m}\sigma}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\ - \frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}&\Bigg(\\ \frac{\partial^{a+1+c+e+m}u}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}}& \\ + \frac{\partial^{a+c+1+e+m}v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}}& \\ + \frac{\partial^{a+c+e+1+m}w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}}\Bigg)\Bigg]&.\end{aligned} \tag{16c}$$

The continuity equation is rearranged to isolate the temporal term in (16a). Equation (16b) is an expression for the nth temporal derivative of the continuity equation, found by applying the Leibniz rule to t-differentiation and (16a). Finally, the Leibniz rule is again used (three times) with (16b) to obtain (16c), the full expression for any x, y, z, t derivative of the continuity equation. The corresponding expressions for the other equations in (12) are given in the Appendix as (50)–(55). Taken together, (16c) and (50)–(55) are an explicit representation of the GET_EXPR and EVAL_EXPR functions in Algorithm 1, i.e., they allow EVAL_EXPR(GET_EXPR(deriv_id), deriv_cache) to be implemented in a computer code without resorting to a CAS.

Equations (16c) and (50)–(55) are called recurrence relations because they express higher-order temporal derivatives of the Navier-Stokes equations in terms of lower-order derivatives. Understanding how this recursive process works is made easier by expressing (12) in the more-general form

$$\frac{\partial U}{\partial t} = f\left(U, \frac{\partial U}{\partial x}, \frac{\partial U}{\partial y}, \frac{\partial U}{\partial z}, \frac{\partial^2 U}{\partial x^2}, \frac{\partial^2 U}{\partial x\,\partial y}, \frac{\partial^2 U}{\partial x\,\partial z}, \frac{\partial^2 U}{\partial y^2}, \frac{\partial^2 U}{\partial y\,\partial z}, \frac{\partial^2 U}{\partial z^2}, \right) \tag{17}$$

where U are the unknowns σ, u, v, w, p. The spatial coordinates are collapsed into one unknown in the expression

$$\frac{\partial U}{\partial t} = f\left(U, \frac{\partial U}{\partial X}, \frac{\partial^2 U}{\partial X^2}, \right) \tag{18}$$

where the notation $\frac{\partial^n(\ )}{\partial X^n}$ stands for any derivative with respect to $x^a y^b z^c$ for all $a+b+c=n$ (so, for example, $\frac{\partial^2 U}{\partial X^2}$ indicates the derivatives $\frac{\partial^2 U}{\partial x^2}$, $\frac{\partial^2 U}{\partial x\,\partial y}$, $\frac{\partial^2 U}{\partial x\,\partial z}$, $\frac{\partial^2 U}{\partial y^2}$, $\frac{\partial^2 U}{\partial y\,\partial z}$, and $\frac{\partial^2 u}{\partial z^2}$). Next, consider each term on the right-hand side of (18). When building the Taylor series in (7), the derivatives will naturally be calculated in increasing order, i.e., first U will be found, then $\frac{\partial U}{\partial t}$, $\frac{\partial^2 U}{\partial t^2}$, etc.. If this is so, then a value for U will already be known when attention is turned to calculating $\frac{\partial U}{\partial t}$, but $\frac{\partial U}{\partial X}$ and $\frac{\partial^2 U}{\partial X^2}$ will not. To express this dependency, the notation

$$\frac{\partial U}{\partial t} \rightarrow \frac{\partial U}{\partial X}, \frac{\partial^2 U}{\partial X^2} \tag{19}$$

will be used, and means "$\frac{\partial U}{\partial X}$ and $\frac{\partial^2 U}{\partial X^2}$ are needed to calculate $\frac{\partial U}{\partial t}$." Equation (19) is expressed graphically in Figure 2.

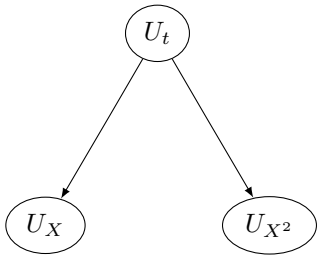

Figure 2. CK recursion dependency diagram for (19)**.**

Next, consider the equivalent of (18) for $\frac{\partial^2 U}{\partial t^2}$. After recognizing that all terms in (12)-(14) are sums of products, one sees that the correct relationship is

$$\frac{\partial^2 U}{\partial t^2} = f\left(U, \frac{\partial U}{\partial t}, \frac{\partial U}{\partial X}, \frac{\partial^2 U}{\partial X\,\partial t}, \frac{\partial^2 U}{\partial X^2}, \frac{\partial^3 U}{\partial X^2\,\partial t}, \right). \tag{20}$$

Compare (20) to (18). If only the new terms in the right-hand side of (20) are retained (i.e., those that are found in (20) but not (18)), then the resulting dependency expression for $\frac{\partial^2 U}{\partial t^2}$ is

$$\frac{\partial^2 U}{\partial t^2} \rightarrow \frac{\partial^2 U}{\partial X\,\partial t}, \frac{\partial^3 U}{\partial X^2\,\partial t}, \tag{21}$$

which is recognized as the same expression that would have been found by differentiating each term in the right-hand side of (19) with respect to t. Knowing this, a general dependency relationship for the temporal and mixed spatial-temporal derivatives of the Navier-Stokes equations can quickly be determined by differentiating (19) with respect to t^n and X^a:

$$\frac{\partial^{a+n+1} U}{\partial X^a\,\partial t^{n+1}} \rightarrow \frac{\partial^{a+1+n} U}{\partial X^{a+1}\,\partial t^n}, \frac{\partial^{a+2+n} U}{\partial X^{a+2}\,\partial t^n} \tag{22}$$

for $a, n \geq 0$.

Equation (22) allows one to create graphical dependency diagrams like Figure 2 for higher-order derivatives. For example, Figure 3 shows the result of beginning with $\frac{\partial^2 U}{\partial t^2}$ and using (22) until only pure-spatial derivatives are left; Figure 4 shows the same for $\frac{\partial^3 U}{\partial t^3}$.

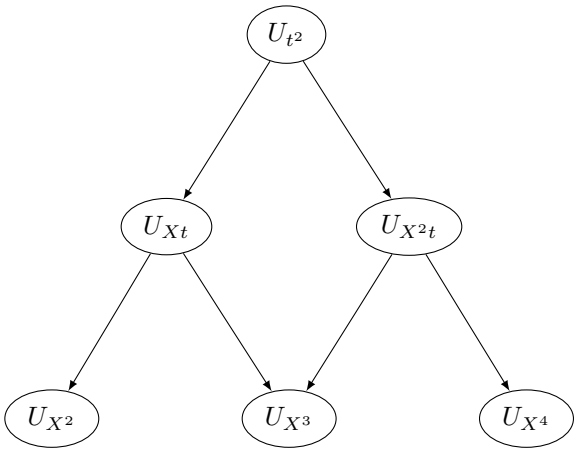

Figure 3. CK recursion dependency diagram for $\frac{\partial^2 U}{\partial t^2}$ and its dependents.

Figures 2–4 are top-down representations of the Cauchy-Kowalewski process — they begin with the desired result ($\frac{\partial^n U}{\partial t^n}$) and end with the input, or starting point ($\frac{\partial^m U}{\partial X^m}$). One would prefer to unwind the

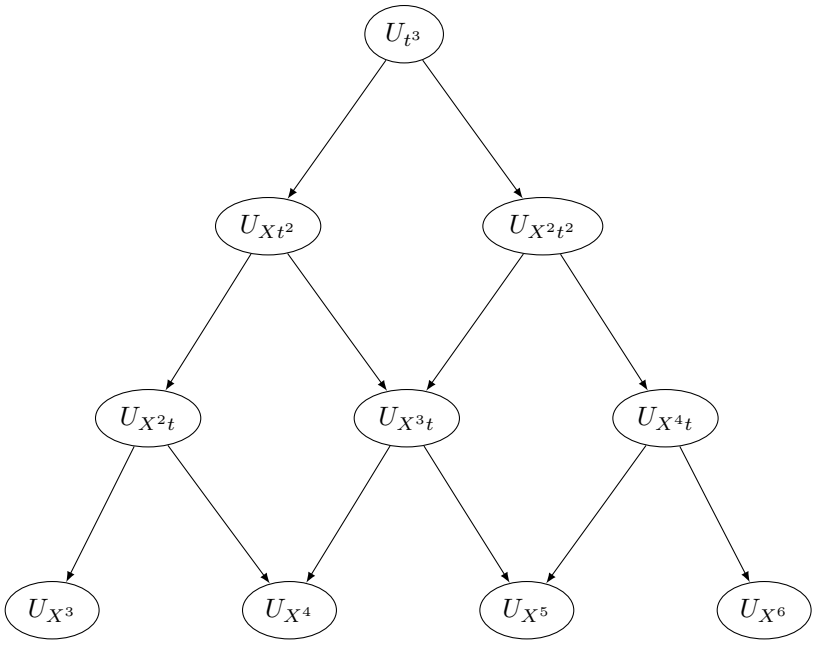

Figure 4. CK recursion dependency diagram for $\frac{\partial^3 U}{\partial t^3}$ and its dependents.

recursion and instead start from the new pure spatial derivatives needed for a given temporal derivative, then find the required mixed spatial-temporal derivatives, then finally the desired pure-temporal derivative. This can be achieved by simply eliminating all but the right two nodes of each row in Figures 2–4, and then moving from left-to-right, bottom-to-top with those that remain. After removing the unnecessary nodes, Figure 3 becomes Figure 5, and Figure 4, Figure 6 (Figure 2 is unchanged).

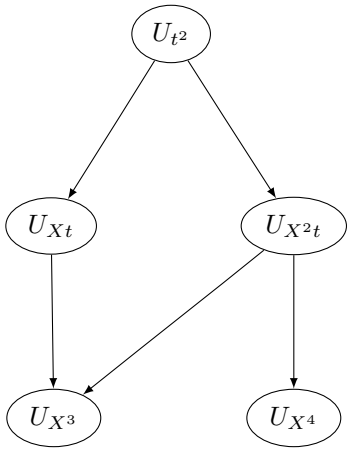

Figure 5. CK recursion dependency diagram for $\frac{\partial^2 U}{\partial t^2}$ and its dependents with unnecessary nodes removed.

The "left-to-right, bottom-to-top" calculation procedure just described is represented in pseudocode in Algorithm 2. Here, the deriv_cache variable is imagined to be some object that holds the numerical values of derivatives calculated previously (identical to its meaning in Algorithm 1), and tot_order is an integer > 0 equal to the order of the pure-temporal derivative that will be calculated by the CK_UNWOUND procedure (e.g., tot_order = 2 for Figure 5 and tot_order = 3 for Figure 6). X_order and t_order are also integer variables, indicating the spatial and temporal order, respectively, of the derivatives that will be calculated by the EVAL_DERIV routine and stored in the deriv_cache variable — for example, if X_order = 2 and t_order = 1 are passed to EVAL_DERIV, then $\frac{\partial^3 U}{\partial X^2 \, \partial t}$ will be found and saved in deriv_cache.

A pseudocode listing of the EVAL_DERIV routine is shown in Algorithm 3. The outer **if**/**else** block

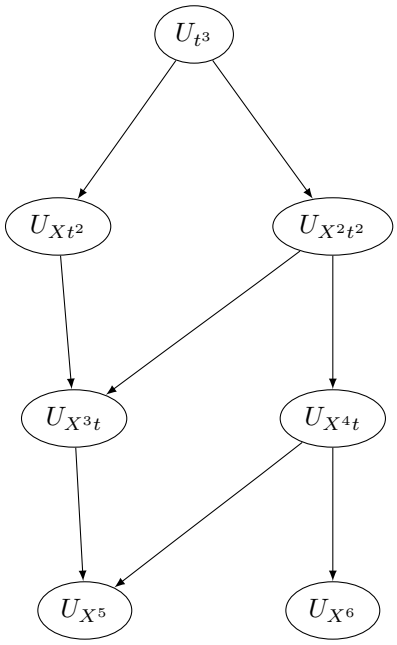

Figure 6. CK recursion dependency diagram for $\frac{\partial^3 U}{\partial t^3}$ and its dependents with unnecessary nodes removed.

Algorithm 2 Pseudocode for unwound Cauchy-Kowalewski recursion with the Navier-Stokes equations.

```
procedure CK_UNWOUND(tot_order, deriv_cache)
    X_order ← 2 · tot_order
    for t_order ← 0, tot_order − 1 do
        X_order ← X_order − 1
        EVAL_DERIV(X_order, t_order, deriv_cache)
        X_order ← X_order + 1
        EVAL_DERIV(X_order, t_order, deriv_cache)
        X_order ← X_order − 2
    end for
    EVAL_DERIV(X_order, t_order, deriv_cache)          ▷ Here X_order = 0, t_order = tot_order
end procedure
```

decides if the desired derivatives are temporal/mixed-spatial-temporal or pure spatial derivatives of the initial condition. The code for the two branches is similar: both contain loops that iterate over the derivatives indicated by the X_order and t_order derivatives (and, therefore, the $\frac{\partial^{a+n}U}{\partial X^a \partial t^n}$ notation), and both eventually use the STORE_VAL routine to save each iteration's work in the deriv_cache variable. The x_order, y_order, and z_order variables are integers that correspond to the x-, y, and z-order of the derivative calculated by either EVAL_RECUR_REL (using the Navier-Stokes recurrence relations in (16c) and (50)–(55)) or EVAL_IC (by differentiating the known initial condition).

Algorithm 3 Pseudocode for EVAL_DERIV in Algorithm 2.

```
procedure EVAL_DERIV(X_order, t_order, deriv_cache)
    if t_order > 0 then
        for z_order ← 0, X_order do
            for y_order ← 0, X_order − z_order do
                x_order ← X_order − z_order − y_order
                deriv_val ← EVAL_RECUR_REL(x_order, y_order, z_order, t_order, deriv_cache)
                STORE_VAL(x_order, y_order, z_order, t_order, deriv_val, deriv_cache)
            end for
        end for
    else
        for z_order ← 0, X_order do
            for y_order ← 0, X_order − z_order do
                x_order ← X_order − z_order − y_order
                deriv_val ← EVAL_IC(x_order, y_order, z_order, deriv_cache)
                STORE_VAL(x_order, y_order, z_order, t_order, deriv_val, deriv_cache)
            end for
        end for
    end if
end procedure
```

Taken together, the recurrence relations (16c), (50)–(55) and the pseudocode in Algorithms 2 and 3 can be used to efficiently implement Cauchy-Kowalewski recursion and build a Taylor series solution (7) to the Navier-Stokes equations. The Taylor series will not, of course, contain an infinite number of terms, and thus involve some truncation error. In this work, an estimate of the truncation error is obtained by adding together the magnitude of the last two Taylor series terms, and then compared to a desired error tolerance set by the user of the EVA tool. If the error estimate is less than the user-specified value, the order of the Taylor series is increased; otherwise, the EVA solution for the current spatial location is deemed adequate and attention is turned to the next grid point.

IV. Results

A. BASS

The CAA code verified in this work is NASA Glenn's Broadband Aeroacoustic Stator Simulator[27] (BASS), a high-order, block-structured finite-difference LES solver. BASS is capable of working with the fully nonlinear, Euler or Navier-Stokes equations in conservative form and two or three dimensions — three-dimensional Navier-Stokes results will be presented here.

A variety of time-marching and spatial differencing schemes are implemented in BASS. The spatial stencils available include the Dispersion Relation Preserving (DRP) scheme of Tam and Webb[28] with coefficients from Tam and Shen,[29] and Hixon's prefactored form[30] of the compact 6^{th}-order scheme of Lele[31] (C_6), as well as standard 2^{nd}- and 6^{th}-order explicit central differencing (E_2 and E_6, respectively). For time-marching, the four- and five-stage Runge-Kutta schemes of Jameson[32] (RK4L and RK5L, respectively), the 5/6-stage two-step Runge-Kutta scheme of Stanescu and Habashi[33] (RK56), and the High-Accuracy Large-step Explicit Runge-Kutta (HALE-RK) 7-stage and 6/7-stage two-step schemes of Allampalli et al.[34] (RK7S and RK67) are used in this work.

B. Verification Procedure

The procedure used to verify the BASS code with EVA is summarized in the following steps:

1. Set up the problem — specify the initial condition, final time level, extent of the computational domain, etc.,
2. Use EVA to find the solution at the final time level,
3. Use BASS to find the solution at the final time level with a range of discretization measures (grid spacings for spatial verification, time step sizes for temporal),
4. Calculate the error in the BASS solutions through comparison with the EVA solution,
5. Calculate the error convergence rate (or observed order-of-accuracy).

In reference to step 4, two different error norms were in this work: the l_2

$$\epsilon_{l_2} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\phi_n^{(i)} - \Phi_n\right)^2}, \tag{23}$$

and the $l_{\max}$

$$\epsilon_{l_{\max}} = \max_n \left|\phi_n^{(i)} - \Phi_n\right|. \tag{24}$$

The notation Φ and $\phi^{(i)}$ indicate the EVA and i-th BASS code solutions, respectively, with the subscripts representing the solution at each of the N spatial locations in common between Φ and ϕ. The $l_{\max}$ norm tends to be more volatile than the l_2.

In step 5, the error convergence rate was calculated using two different methods. The first approach assumed the BASS solution error could be well-approximated by the expression

$$\epsilon_i = \phi_i - \Phi \approx Ah_i^p. \tag{25}$$

To find the convergence rate p, the ratio of the error of two PDE code solutions is formed

$$\frac{\epsilon_i}{\epsilon_{i-1}} \approx \frac{Ah_i^p}{Ah_{i-1}^p} = \frac{h_i^p}{h_{i-1}^p} = \left(\frac{h_i}{h_{i-1}}\right)^p \tag{26}$$

and then rearranged to isolate p,

$$p \approx \frac{\log\left(\frac{\epsilon_i}{\epsilon_{i-1}}\right)}{\log\left(\frac{h_i}{h_{i-1}}\right)} = \frac{\log\left(\epsilon_i\right) - \log\left(\epsilon_{i-1}\right)}{\log\left(h_i\right) - \log\left(h_{i-1}\right)}. \tag{27}$$

The second expression for p in (27) shows the connection between the convergence rate of the error and the error itself: the former is the slope of the latter on a log-log plot.

The second convergence rate calculation method makes a slightly more general assumption about the BASS error, specifically

$$\epsilon_i \approx Ah_i^p + B, \tag{28}$$

where B is a part of the error insensitive to the discretization measure being refined (grid spacing or time step). Equation 28 contains three unknowns (A, B, and p), and each ϵ-h combination is associated with one BASS code calculation — thus error norm data from three runs will be needed to solve for the unknowns in (28), i.e.,

$$\begin{aligned}
\epsilon_0 &= Ah_0^p + B \\
\epsilon_1 &= Ah_1^p + B \\
\epsilon_2 &= Ah_2^p + B,
\end{aligned} \tag{29}$$

which is a non-linear system with no explicit solution. Following Oberkampf and Roy,[1] the unknowns A and B may be eliminated with a bit of manipulation, yielding

$$\frac{h_0^p - h_1^p}{h_1^p - h_2^p} = \frac{\epsilon_0 - \epsilon_1}{\epsilon_1 - \epsilon_2}, \tag{30}$$

one equation for the unknown p. Unless (as in Oberkampf and Roy) $h_i = r^i h_0$, Equation (30) cannot be solved explicitly and a root-finding algorithm must be used to find p — the approach taken here.

C. Test Case Parameters

For the present test cases, the initial distribution of the primitive flow variables were of the form

$$\begin{aligned}\phi(x,y,z) =& \overline{\phi} + \widetilde{\phi} \exp\left(-\frac{\ln 2}{0.15^2}\left[(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2\right]\right) \\ & \sin\left(k_x x + k_y y + k_z z + \theta\right),\end{aligned} \tag{31}$$

with the values of the various constants given in Table 1. The location of the Gaussian perturbations peaks (i.e., x_0, y_0, z_0 from (31)) for each of the flow variables were placed at different locations in the domain to avoid any symmetry in the initial flow; Figure 7 uses colored spheres to represent these location. The values of k_x, k_y, and k_z were set such that the direction the sin() term varies the most rapidly differs for each variable, but has the same wavelength of 0.25, giving an approximate wavenumber $\widetilde{k} = \frac{2\pi}{0.25} = 8\pi$. Figures 8 and 9 show slices of the density and z-momentum initial conditions, respectively.

ϕ	$\overline{\phi}$	$\widetilde{\phi}$	x_0	y_0	z_0	k_x	k_y	k_z	θ
σ	1.	0.001	-0.1	-0.1	-0.1	0.875917	0.0459049	25.1174	7.5
u	0.03	0.006	-0.05	0.05	0.05	4.00477	14.9460	19.8048	9.
v	0.02	0.004	0.05	-0.05	-0.05	-20.2615	13.1580	6.92752	10.5
w	0.01	0.005	-0.05	0.05	-0.05	-18.3538	-14.8626	-8.59590	12.
p	$\frac{1}{\gamma}$	0.01	0.1	0.1	0.1	5.03652	-13.1205	-20.8359	13.5

Table 1. Values of constants for the initial condition found in (31).

The common ratio of specific heats value for air, $\gamma = 1.4$, was used, which, when combined with the choices $\overline{p} = \frac{1}{\gamma}$ and $\overline{\rho} = 1$ give a far-field acoustic velocity of unity. The maximum propagation speed based on far-field quantities, then, would be $c = 1 + \sqrt{0.03^2 + 0.02^2 + 0.01^2} \approx 1.0374$. A dynamic viscosity $\mu = 0.0002$ was chosen, which, when combined with a reference length $L = 2 \cdot 0.15$ (twice the Gaussian half-width in the initial condition), $V = \sqrt{0.03^2 + 0.02^2 + 0.01^2}$ (far-field velocity), and $\overline{\rho} = 1$, gives a Reynolds number

$$Re = \frac{\overline{\rho} V L}{\mu} \approx 56.1, \tag{32}$$

much lower than what is typically found in aerospace applications. The rationale behind using such a small value is related to the usual physical interpretation of the Reynolds number: since it represents the ratio of the inviscid and viscous terms, a Reynolds number closer to unity indicates these terms are approximately the same magnitude, and problems with one is unlikely to be hidden by the other. Likewise, the Prandtl number Pr, often described as the ratio of the viscous and conductive terms, was set to 0.7 (the typical value for air). Finally, the Peclet number, defined $Pe = Re \cdot Pr$ and a measure of the ratio of the inviscid and conductive terms, was $Pe = 56.1 \cdot 0.7 = 39.3$.

Because EVA currently solves the Navier-Stokes equations on an infinite domain, the enforcement of a particular boundary condition along surfaces in the solution domain is not possible. BASS, like all PDE codes, requires boundary conditions on all domain boundaries, however, and this represents a potential discrepancy between the EVA and BASS solutions. To minimize the possibility of mismatched flow solutions at the BASS domain boundaries from negatively impacting the verification results, the Giles non-reflecting boundary condition[35] and an initial condition with a limited spatial extent (see Figure 7) was used.

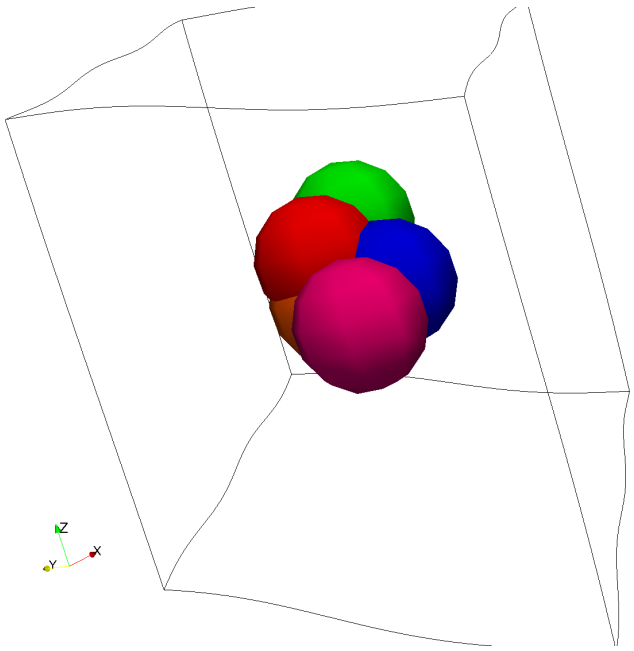

Figure 7. Outline of the BASS verification test case grid with spheres marking the peaks of the Gaussian perturbations of the initial flow, with purple, red, blue, orange, and green indicating the σ, u, v, w, and p centers, respectively. The radius of each sphere is 0.15, the half-width of the Gaussians.

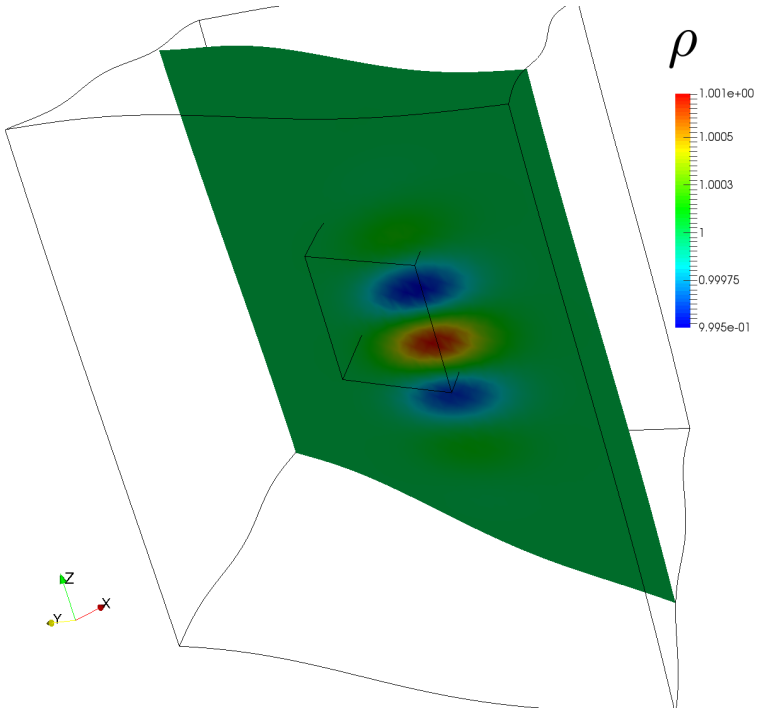

Figure 8. Slice of the test case grid colored by the ρ initial condition from Equation (31). The inner box indicates the region where BASS's solution was compared with EVA.

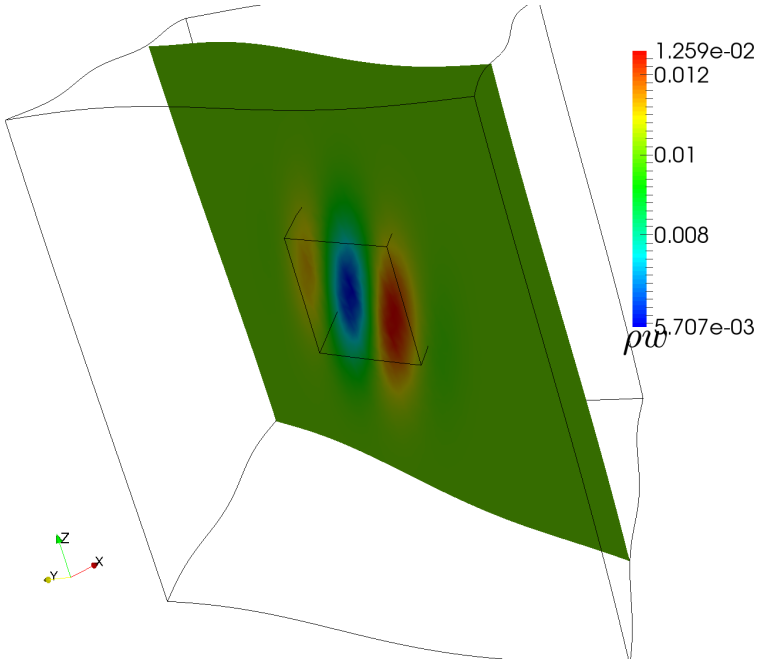

Figure 9. Slice of the test case grid colored by the ρw initial condition from Equation (31). The inner box indicates the region where BASS's solution was compared with EVA.

D. Spatial differencing scheme verification

1. *Linear analysis*

In this section, the results of verifying the implementation of BASS's spatial difference schemes with the EVA tool will be presented. While not required to verify a numerical method, analysis of the accuracy of a scheme is often helpful in interpreting code verification results. The accuracy of both spatial differencing and time-marching schemes are commonly analyzed using Fourier analysis. For spatial differencing schemes, this approach consists of investigating the error associated with approximating the derivative of a single component of a Fourier series $f_k(x) = e^{ikx}$, where k is a real constant (the wavenumber) and $i = \sqrt{-1}$. Using the second-order central scheme

$$D_x(f(x,t)) = \frac{f(x+\Delta x,t) - f(x-\Delta x,t)}{2\Delta x} \tag{33}$$

to differentiate $f_k(x)$ gives

$$\begin{aligned} D_x(f_k(x)) &= \frac{f_k(x+\Delta x) - f_k(x-\Delta x)}{2\Delta x} \\ &= \frac{1}{2\Delta x}\left(e^{ik[x+\Delta x]} + e^{ik[x-\Delta x]}\right) \\ &= \frac{i}{\Delta x}\left[\sin(k\Delta x)\right] e^{ikx}, \end{aligned} \tag{34}$$

which is then compared to the exact value

$$\frac{\partial f_k(x)}{\partial x} = ike^{ikx} = \frac{i}{\Delta x}\left[k\Delta x\right] e^{ikx}. \tag{35}$$

The bracketed quantity in (34),

$$(k\Delta x)^* = \sin(k\Delta x), \tag{36}$$

defined as the numerical wavenumber, appears when evaluating the relative error

$$\epsilon\left(k\Delta x\right) = \frac{D_x(f_k) - \frac{\partial f_k}{\partial x}}{\frac{\partial f_k}{\partial x}} = \frac{(k\Delta x)^*}{k\Delta x} - 1 \tag{37}$$

associated with a spatial differencing scheme. Also of interest is p, the convergence rate of the relative error, which can be calculated by considering the slope of $\epsilon(k\Delta x)$ on a log-log plot, i.e.,

$$\begin{aligned} p &= \frac{\mathrm{d}\log(\epsilon)}{\mathrm{d}\log(k\Delta x)} \\ &= \frac{\mathrm{d}\log(\epsilon)}{\mathrm{d}k\Delta x}\frac{\mathrm{d}k\Delta x}{\mathrm{d}\log(k\Delta x)} \\ &= \frac{\mathrm{d}\log(\epsilon)}{\mathrm{d}k\Delta x}k\Delta x \\ &= \frac{1}{\epsilon}\frac{\mathrm{d}\epsilon}{\mathrm{d}k\Delta x}k\Delta x. \end{aligned} \tag{38}$$

The numerical wavenumber, relative error, and convergence rate of the spatial differencing schemes available in BASS are shown in Figure 10 and the top and bottom of Figure 11, respectively. The latter plot uses the number of points-per-wavelength $\frac{2\pi}{k\Delta x}$ for the abscissa.

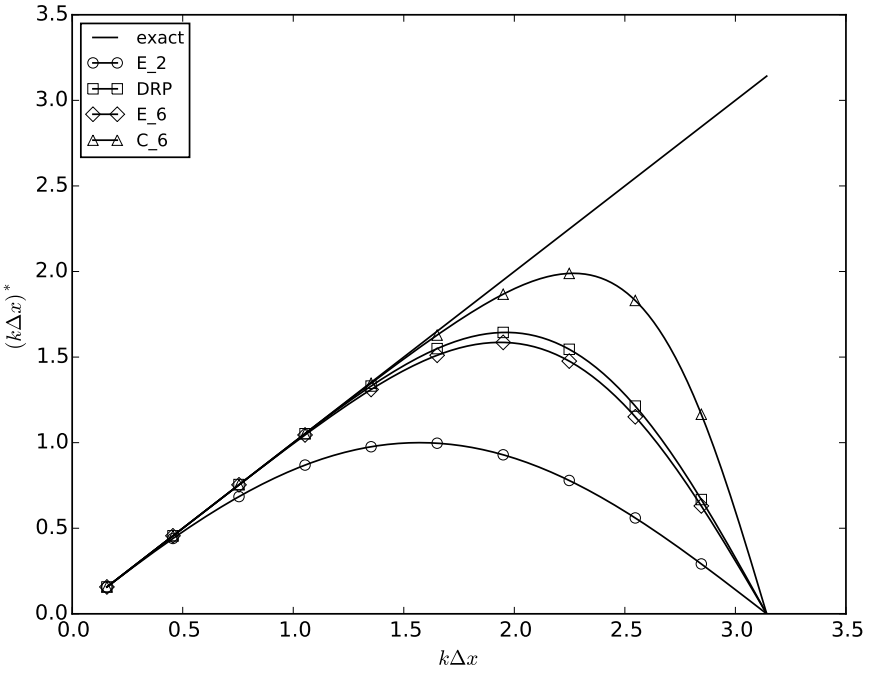

Figure 10. Numerical wavenumber $(k\Delta x)^*$ of the four spatial differencing schemes implemented in BASS.

A few features of Figures 10 and 11 are worth mentioning. For the numerical wavenumber results in Figure 10, the performance of each scheme can be judged by how closely it follows the curve corresponding to the exact value $(k\Delta x)^* = k\Delta x$. Clearly, then, the explicit second-order and compact sixth-order schemes are the least and most accurate, respectively. The DRP and explicit sixth-order schemes make for a more interesting comparison, however: the DRP scheme outperforms the explicit sixth-order for higher $k\Delta x$ values, but it becomes difficult to distinguish the two schemes as $k\Delta x$ decreases. Inspection of the relative error plot in Figure 11, however, plainly shows that the DRP scheme's error is actually higher than the explicit sixth-order for high points-per-wavelength (low $k\Delta x$). The DRP scheme is an optimized form of the explicit sixth-order: it is essentially the explicit sixth-order scheme with modified coefficients that give better performance for moderate points-per-wavelength.

The convergence rate curves clearly identify the order of the truncation error of each scheme: the explicit second-order scheme is second-order, the DRP scheme is fourth-order, and the explicit and compact sixth-order schemes are sixth-order. Considering the top and bottom plots of Figure 11 helps illustrate the difference between accuracy and order-of-accuracy. For example, despite its lower order-of-accuracy, the

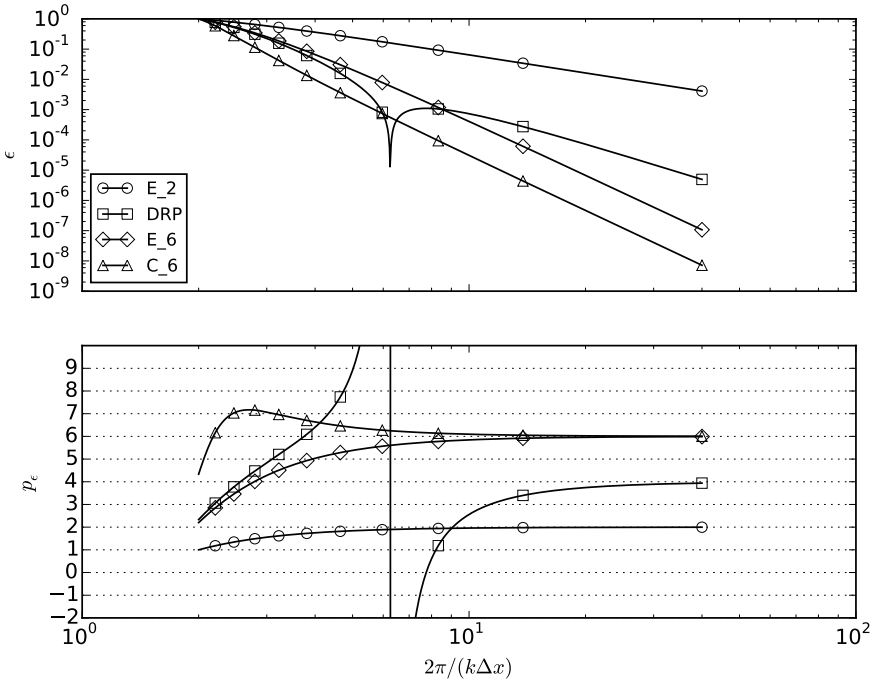

Figure 11. Relative error and convergence rate of the four spatial differencing schemes implemented in BASS.

DRP scheme is more accurate than the explicit sixth-order scheme for a considerable range of points-per-wavelength, and even outperforms the very accurate compact sixth-order scheme for a narrow band of $\frac{2\pi}{k\Delta x}$. And though they possess identical orders-of-accuracy, the explicit and compact sixth-order schemes return widely different error over nearly the entire range of $k\Delta x$ shown in Figure 11, with the compact scheme beating the explicit by about an order-of-magnitude.

2. BASS results

As discussed above, the goal of code verification is to compare the convergence rate of the error of a PDE code to the formal order-of-accuracy of the code's numerical method. For a spatial differencing scheme, this involves observing how the error behaves as the grid is refined. For the present test cases, each grid began as a uniform Cartesian grid extending from -0.5 to 0.5 in the three coordinate directions, and was then rotated, skewed, and subjected to sinusoidal perturbations. The coarsest grid in the series used 9^3 points — denser grids were constructed by uniformly refining the base grid through the addition of 1, 2, 3... points between each of the original grid's, giving a grid series of 9^3, 17^3, 25^3, 33^3,..., 201^3 points with approximate grid spacing

$$\widetilde{\Delta x} = \frac{1}{\sqrt[3]{N-1}} \tag{39}$$

ranging from $\frac{1}{8}$ to $\frac{1}{200}$. Figure 12 shows the 41^3-point example from this grid series.

For each BASS run, the RK7S time-marching scheme was used to march the initial flow to a final time level of $t = 0.0005$ with one time step. The choices of very small Δt and highly accurate time-marching scheme are meant to ensure that the error associated with the time-marching scheme is much lower than that of the spatial differencing. For the EVA calculation, a desired truncation error of 10^{-15} was used, which was easily satisfied, again thanks to the small time step size.

The l_2 error and the convergence rate of the l_2 norm, p_{l_2}, for the density flow variable, are shown in Figure 13. The quantity $2\pi/(\widetilde{k\Delta x})$, an approximation of the number of points-per-wavelength, is used for the abscissa to facilitate comparison with Figure 11. While not identical, Figures 11 and 13 are remarkably similar. Like the linear analysis, the compact sixth-order and explicit second-order schemes are clearly the

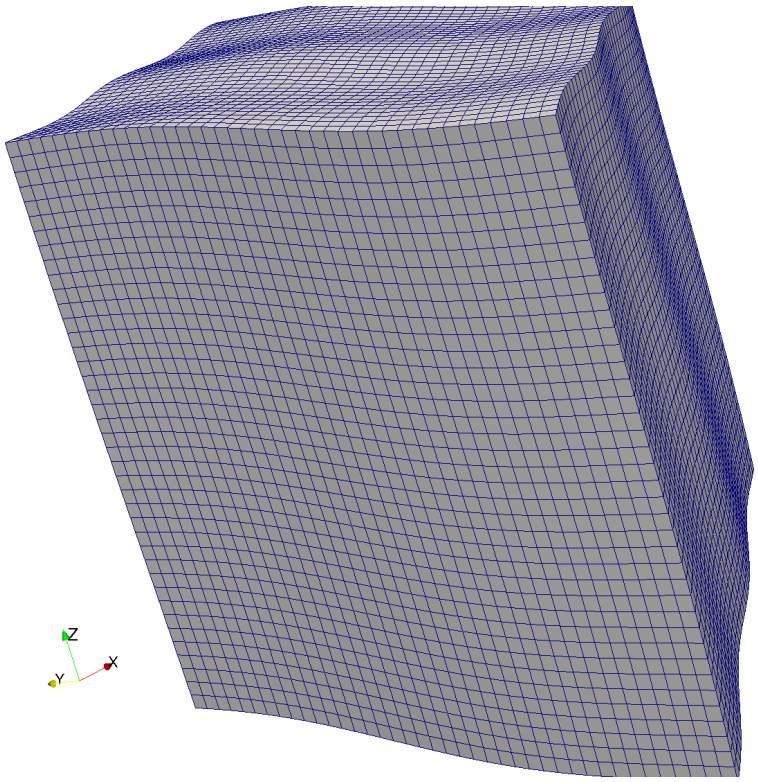

Figure 12. BASS verification test case grid with 41^3 points.

most and least accurate, respectively. The DRP scheme outperforms the explicit sixth-order scheme for moderate points-per-wavelength, but then returns higher error for the high $2\pi/(\widetilde{k\Delta x})$ values, as predicted by Figure 11. Many similarities are found in the convergence rate results, also: the compact sixth scheme initially shows higher-order behavior than the explicit sixth; the characteristic order spike is seen in the DRP curve, and the explicit second-order converges at its expected rate for a wide range of points-per-wavelength.

Overall, it made little difference if the convergence rate was calculated using (27) or (30). Figure 14 compares the two approaches for the l_{max} norm of the ρu error. The expected convergence rate is eventually observed for each scheme, and the curves are qualitatively similar. This indicates that the temporal component of the error is small relative to that of the spatial differencing scheme, and thus the assumption of (25) is a good one.

Achieving asymptotic convergence is the most difficult for the compact sixth-order scheme. Figure 15 compares the compact and explicit sixth-order performance for all flow variables. Notice how significantly more points-per-wavelength are required for the compact scheme to achieve the expected p, a result not predicted by the linear analysis shown in Figure 11. The cause may be related to the Giles non-reflecting boundary conditions used by the BASS code: any discrepancy between these BCs and the EVA solution will be propagated more readily by the compact scheme, as it requires a global matrix solve along each grid line. One would expect errors from the boundaries to become less significant as the number of grid points increases, which corresponds well with the behavior seen in Figure 15.

All of the test cases discussed thus far have involved serial BASS calculations, i.e., the code used just one processor to compute the entire flow field. BASS, however, is a parallel code, and can decompose the solution domain into blocks that are distributed to and solved by many processors. To investigate the portions of the code responsible for the parallelization, the grid in Figure 12 was split into $8^3 = 512$ blocks, and then run with the same initial condition, grid spacings, and time step sizes as the preceding test cases with 24 processors. Figure 16 displays the grid's block boundaries colored by block index number, showing how multiple blocks were placed inside the EVA domain to ensure any problems with BASS's parallel routines would be noticed.

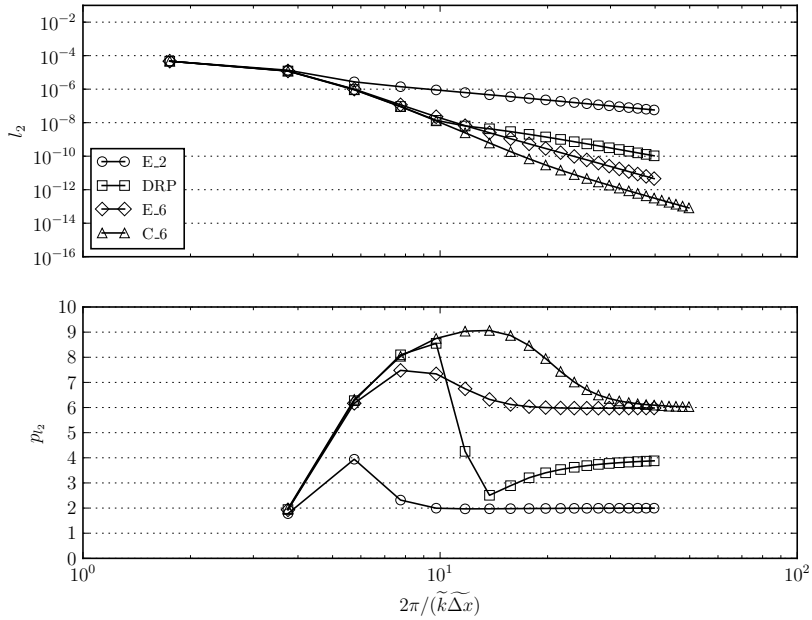

Figure 13. $\rho\ l_2$ **error and convergence rate for each of BASS's spatial differencing schemes.** p_{l_2} **was calculated using** (27)**.**

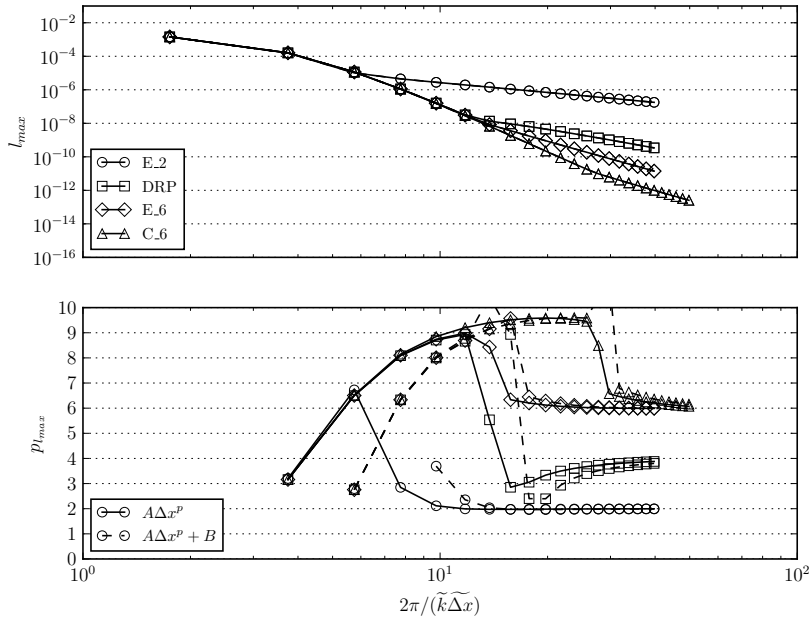

Figure 14. $\rho u\ l_{\max}$ **error and convergence rate for each of BASS's spatial differencing schemes, comparing convergence rate calculation methods** (27) **and** (30)**.**

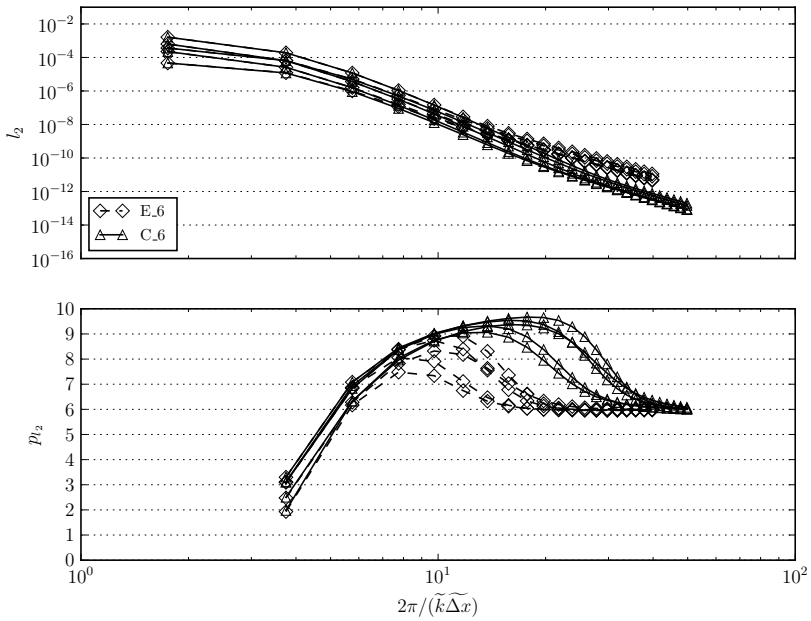

Figure 15. l_2 **error and convergence rate for BASS's sixth-order schemes, with** (27) p_{l_2} **calculation.**

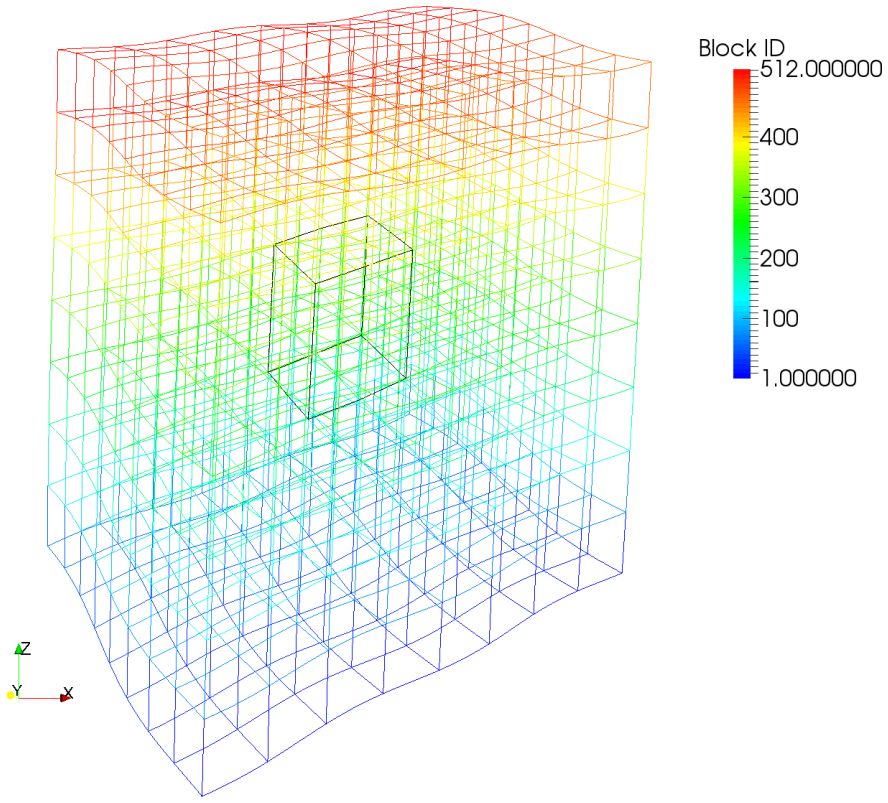

Figure 16. Multiblock grid used for parallel test cases colored by block index. The extent of the EVA solution domain is outlined in black.

Figure 17 shows the density l_2 error and convergence rate results for each of BASS's spatial differencing schemes (the equivalent of Figure 13). The smaller points-per-wavelength grids from the serial test case were not run because they did not meet the minimum points-per-block required by BASS. Like the previous test cases, each scheme's convergence rate eventually attains its expected value.

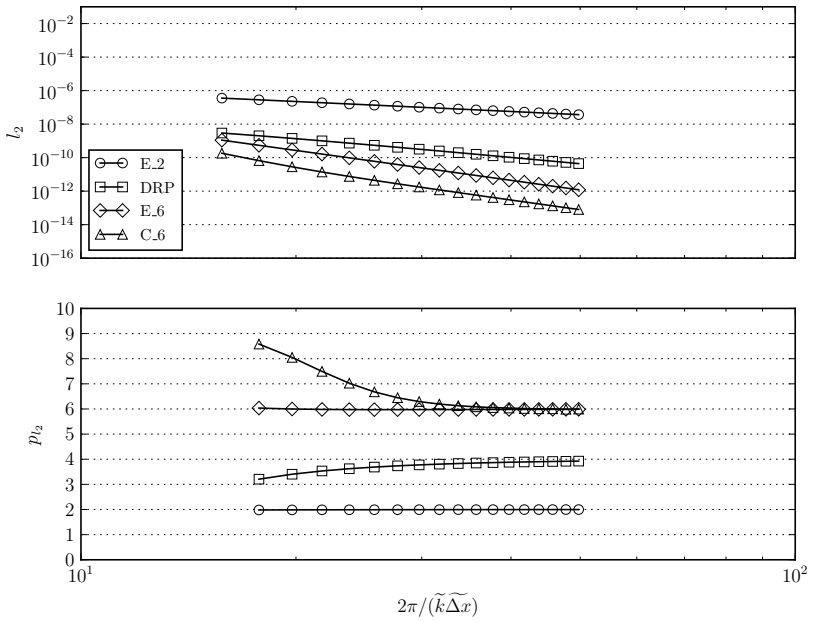

Figure 17. ρ l_2 error and convergence rate for each of BASS's spatial differencing schemes. Parallel test case.

The close agreement between the parallel and serial results in Figure 17 and Figure 13 is unsurprising, since BASS's numerical methods do not change when the code is run in parallel. The one exception is the compact sixth scheme, which uses an explicit sixth-order stencil at interior block boundaries. As described in,[30] this stencil has been tuned to give similar accuracy performance to the compact sixth scheme, and the results shown in Figure 18 confirm that this is the case.

E. Time-marching scheme verification

1. Linear analysis

Here, the results of using EVA to verify BASS's time-marching schemes are presented. As in the spatial differencing scheme verification section, Fourier analysis aids in the interpretation of the verification data, which, when applied to time-marching schemes, usually involves solving a simple ordinary differential equation like

$$\frac{du}{dt} = -i\omega u \tag{40}$$

with exact solution

$$u(t) = u_0 e^{-i\omega t}, \tag{41}$$

where $u_0 = u(t = 0)$ and the time-marching scheme to be analyzed is used to solve (40). Following this procedure with time step size Δt for, say, Euler's first-order explicit scheme

$$u(x, t_0 + \Delta t) = u(x, t_0) + \Delta t \left.\frac{\partial u}{\partial t}\right|_{x,t_0} \tag{42}$$

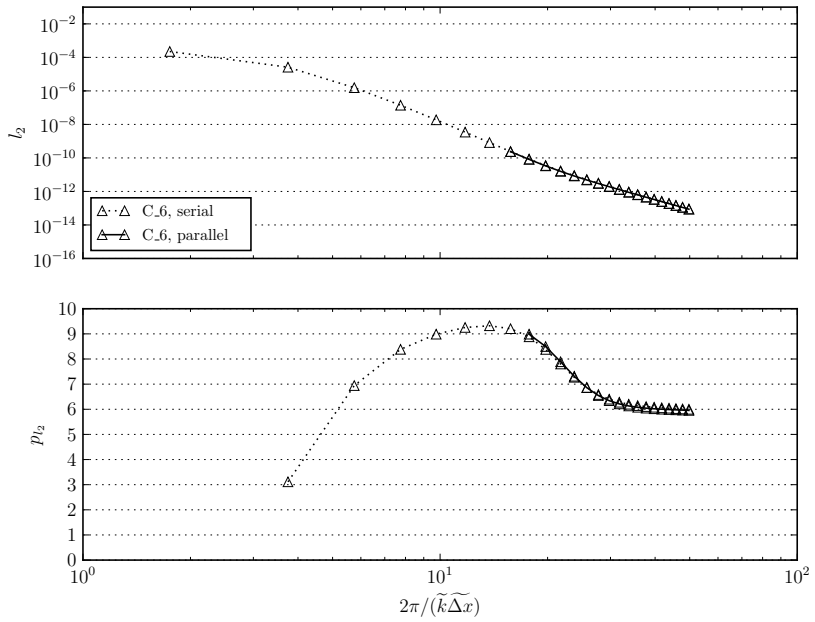

Figure 18. ρv l_2 **error and convergence rate for the compact sixth-order scheme for the serial and parallel test cases.**

gives

$$\begin{aligned} u(\Delta t) &= u_0 + \Delta t\,(-i\omega u_0) \\ &= (1 - i\omega\Delta t)\,u_0 \\ &= G(\omega\Delta t)u_0. \end{aligned} \tag{43}$$

$G = \frac{u(\Delta t)}{u(0)}$, the ratio of the solution at the next and current time step, is called the amplification factor and is analogous to the $(k\Delta x)^*$ for spatial differencing schemes. Once a particular scheme's G is known, the relative error associated with one time step (called the local error, Cf. [36, p. 66]) can be found through the expression

$$\begin{aligned} \theta(\omega\Delta t) &= \frac{u_{\text{scheme}}(\Delta t) - u_{\text{exact}}(\Delta t)}{u_{\text{exact}}(\Delta t)} \\ &= G(\omega\Delta t)/g(\omega\Delta t) - 1 \end{aligned} \tag{44}$$

where $g(\omega\Delta t) = e^{-i\omega\Delta t}$ is the amplification factor for the exact solution (41). The convergence rate p for the error in (44) is found in a manner similar to a spatial differencing scheme's, namely,

$$p = \frac{1}{\theta}\frac{\mathrm{d}\theta}{\mathrm{d}\omega\Delta t}\omega\Delta t. \tag{45}$$

The magnitude of the amplification factor G for each of BASS's schemes is shown in Figure 19, and the corresponding relative local error and convergence rate curves for these schemes are shown in Figure 20. Similar to the $(k\Delta x)^*$ plot of Figure 10, the accuracy of a scheme in Figure 19 is determined by how closely it matches the exact value of the amplification factor magnitude, which for (41) is 1. Figure 19 indicates the stability limit of the schemes, which is the largest value of $\omega\Delta t$ where $|G| \leq 1$, i.e., the amplitude of the solution does not grow after a time step. The amplification factor curves show, then, that the RK56 has both the highest accuracy and most restrictive stability limit of all of the schemes. The RK4L has approximately the same stability limit of the RK56, but dampens the solution considerably. The RK5L is clearly the least accurate scheme, but is much more stable than the RK4L and RK56. Finally, the RK67 and RK7S appear to possess the highest stability limits.

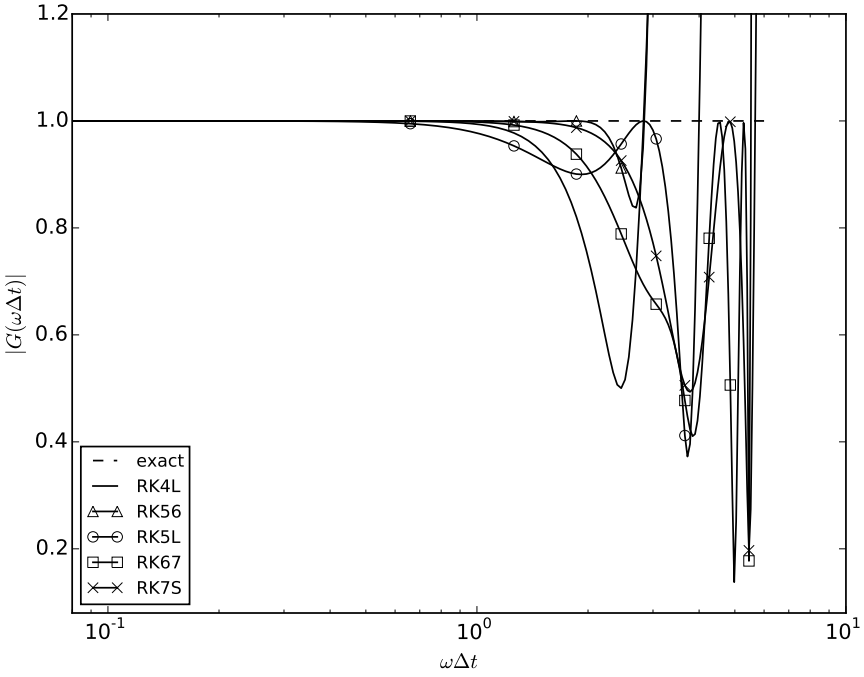

Figure 19. Magnitude of the amplification factor for each of the five time-marching schemes implemented in BASS.

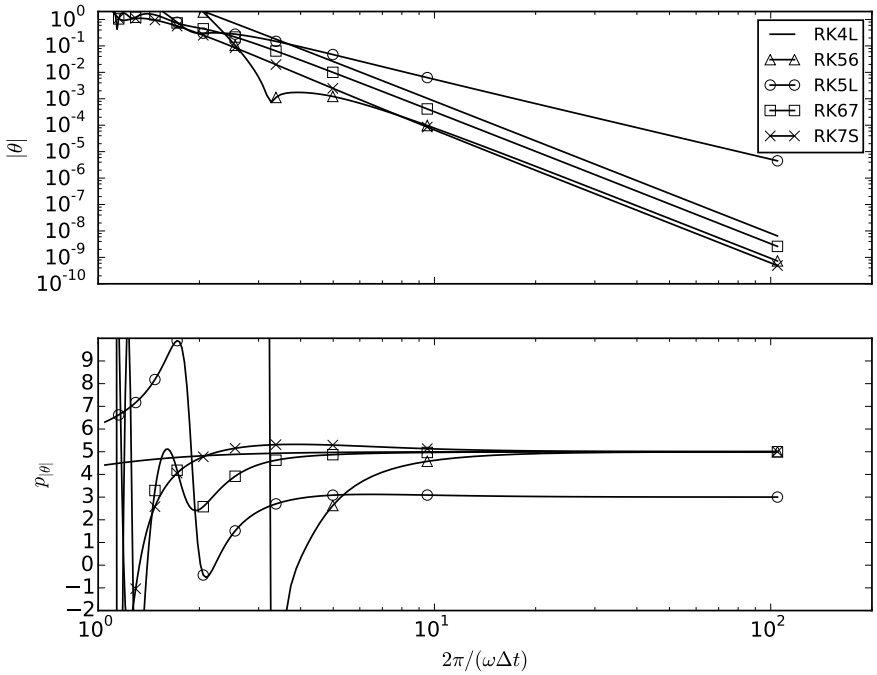

Figure 20. Magnitude of the relative local error and convergence rate for each of the five time-marching schemes implemented in BASS.

Figure 20 allows for a better comparison of the performance of the time-marching schemes. Like the corresponding spatial differencing error plot presented above, the quantity $2\pi/(\omega\Delta t)$, the number of time steps per period of the solution in (41) is used for the abscissa. The error curves confirm that the RK56 is a highly-accurate scheme, showing the sharp error "dip" characteristic of optimized numerical methods (e.g. the DRP spatial differencing scheme in Figure 11), and that the RK5L is by far the least accurate. The convergence rate curves show that the RK5L scheme eventually exhibits third-order behavior, while all of the other schemes converge at a fifth-order rate. The convergence rate of a Runge-Kutta's local error is always one greater than its formal order-of-accuracy, indicating that the RK5L is second-order, and all other schemes are fourth-order. The RK4L scheme, however, is actually fourth-order for linear problems only. This scheme, the classic RK4 scheme (Cf. [36, p. 98, (235i)]) recast as a 2N Runge-Kutta method, drops to second-order when used to solve non-linear ODEs.

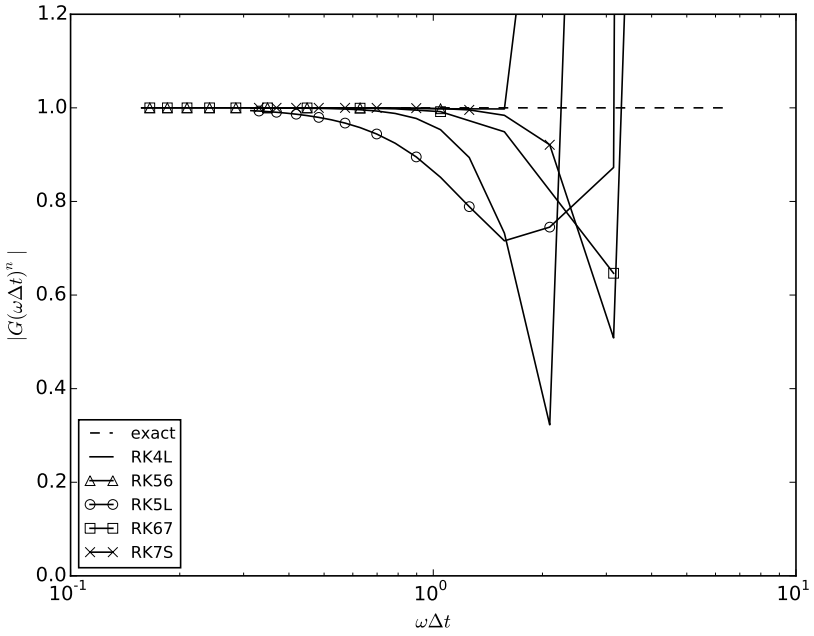

Figure 21. Magnitude of the global amplification factor for each of the five time-marching schemes implemented in BASS after marching to $\omega T = 2\pi$.

Equation (44) shows how the error committed during one time step varies with Δt. Alternatively, one can calculate the error accumulated after marching to a constant final time level $T = mn\Delta t$ using n time steps for a m-step scheme (the global error), which is

$$\begin{aligned}\vartheta(n) &= \frac{u_{\text{scheme}}(T) - u_{\text{exact}}(T)}{u_{\text{exact}}(T)} \\ &= \frac{\left[G\left(\omega\frac{T}{mn}\right)\right]^n}{g(\omega T)} - 1.\end{aligned} \tag{46}$$

An expression for the global error's convergence rate can be found by substituting $\omega\Delta t = \omega T/(mn)$ into (45), which eventually gives

$$p = -\frac{n}{\vartheta}\frac{\mathrm{d}\vartheta}{\mathrm{d}n}. \tag{47}$$

Figures 21 and 22 show the magnitude of the global amplification factor, error, and convergence rate. Again, the RK56 and RK5L are seen to be the most and least accurate, respectively. The global convergence rate results show that each scheme converges at its formal order-of-accuracy. Also note how each scheme's asymptotic range (the range of $\omega\Delta t$ the scheme converges at the expected rate) differs, with the RK56's starting at the highest points-per-cycle value. Typical of low-order methods, the RK5L achieves its design order-of-accuracy more quickly than the other schemes.

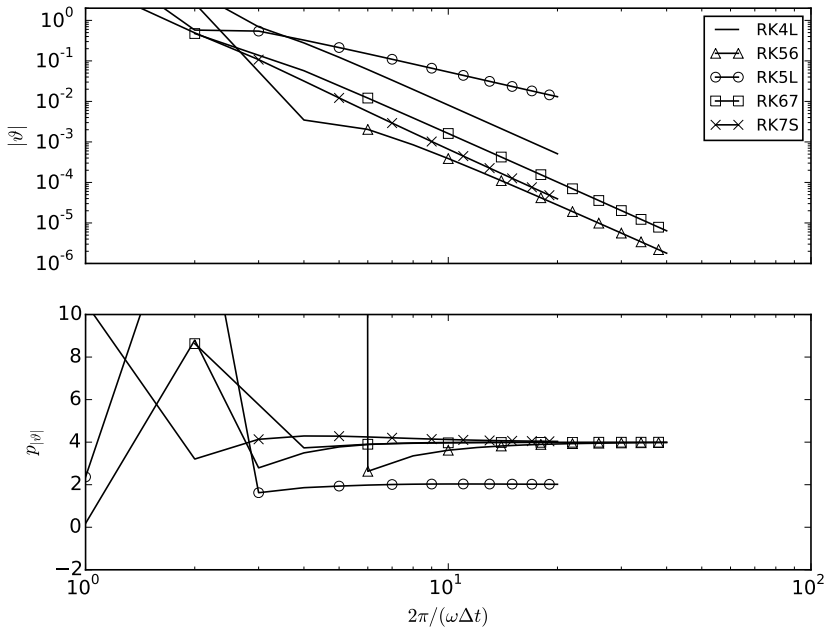

Figure 22. Magnitude of the relative global error and convergence rate for each of the five time-marching schemes implemented in BASS after marching to $\omega T = 2\pi$.

2. BASS results

To investigate the accuracy of BASS's time-marching schemes, the procedure outlined in B was used with the initial condition found in (31) and Table 1, a single 201^3-point grid with the same shape as the example shown in Figure 12, and the explicit sixth-order spatial differencing scheme. BASS was marched to a constant final time level of $t = 0.05$ with 1 to 40 time steps, with the EVA tool provided the reference solution at the same final time level with a target truncation error of 10^{-12}, requiring a 26^{th}-order Taylor series.

Similar to the preceding spatial verification results, an approximate frequency $\widetilde{\omega}$ is used to plot the error and convergence rates as functions of time steps per period of the disturbance. Here, the choice $\widetilde{\omega} = c\widetilde{k}$ was made, where c and $\widetilde{k}$ are the approximate maximum propagation speed and wavenumber from the spatial verification cases, respectively.

Figure 23 shows the density l_2 error and convergence rate results for the temporal test case, with the convergence rate calculated from (27). The high-frequency portion of the plot (i.e., the left side) resembles the global error plot from the linear analysis, Figure 22: the RK56 is the most accurate, followed closely by the RK7S and then RK67, with the RK4L and RK5L showing significantly higher error. The results for the low-wavenumber runs, however, do not line up well with Figure 22: with the exception of the RK5L, the schemes' error curves flatten out, indicating that the error in the BASS calculation is no longer sensitive to the time step size. This, of course, is reflected in the convergence rate results. The RK7S and RK67 schemes only briefly attain their design convergence rates, and the RK56 never does. Also, the RK4L, a second-order scheme for non-linear problems, behaves like a fourth-order scheme before encountering the "error floor."

The density l_2 data from Figure 23 is reproduced in Figure 24, but the convergence rate is calculated using (30), which, unlike the spatial verification results, has a significant impact on the p data. The three fourth-order non-linear schemes now convincingly converge at a fourth-order rate, and the RK5L's error still shows strong second-order behavior. The RK4L scheme's error convergence, however, remains something between third and fourth-order.

The RK4L's convergence rate results in Figure 24 would seem to indicate that the present test case does not contain sufficient nonlinearity for the scheme to converge at its non-linear order-of-accuracy. To determine if this is the case, the RK4L scheme was run with an initial condition identical to that shown in (31) and Table 1, but with larger Gaussian perturbation amplitudes, specifically $\widetilde{\sigma} = 0.1$, $\widetilde{u} = 0.012$,

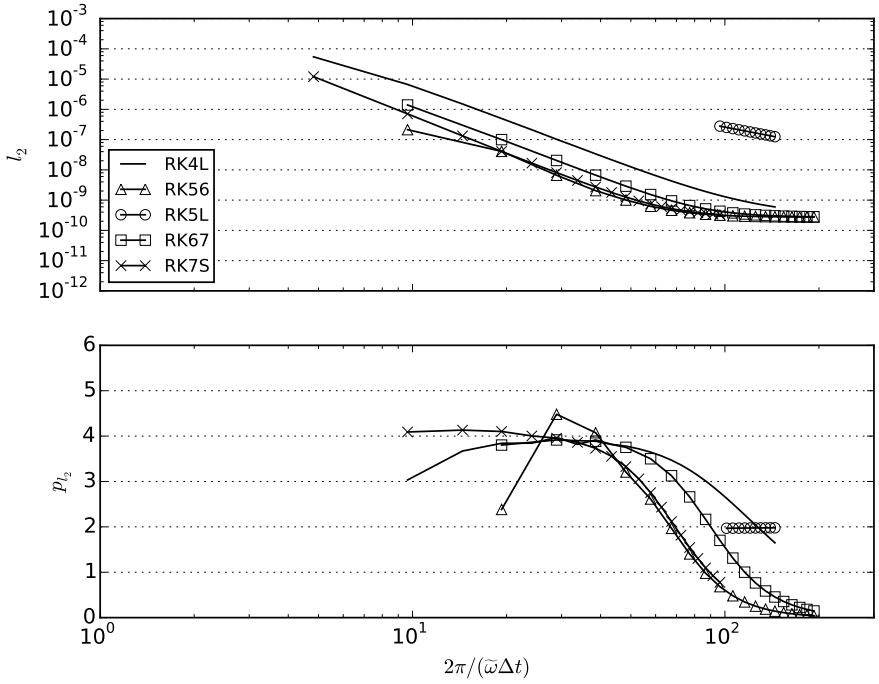

Figure 23. $\rho\, l_2$ **error and convergence rate for each of BASS's time-marching schemes. Equation** (27) **was used to calculate** p_{l_2}**.**

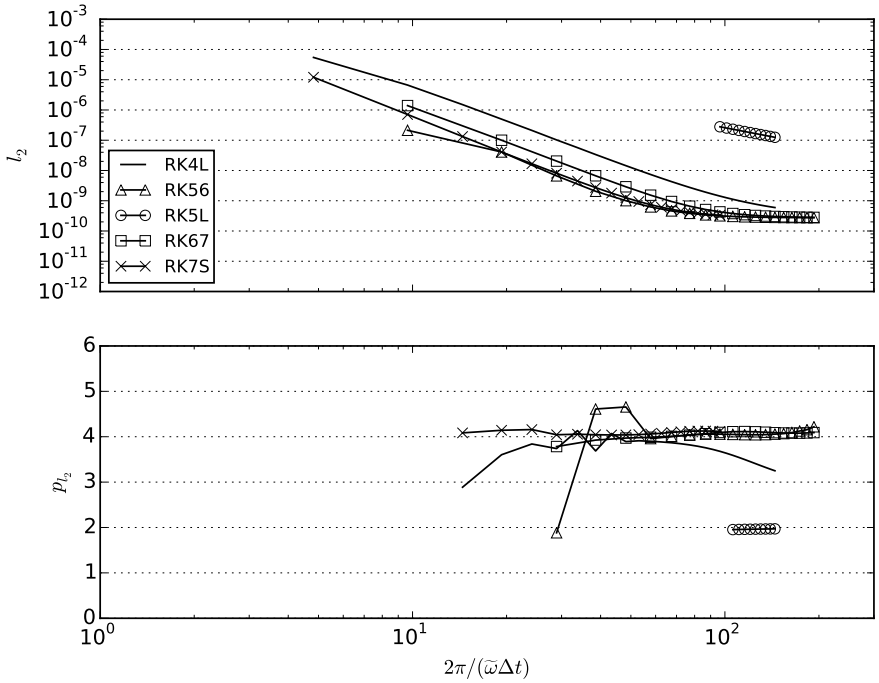

Figure 24. $\rho\, l_2$ **error and convergence rate for each of BASS's time-marching schemes. Equation** (30) **was used to calculate** p_{l_2}**.**

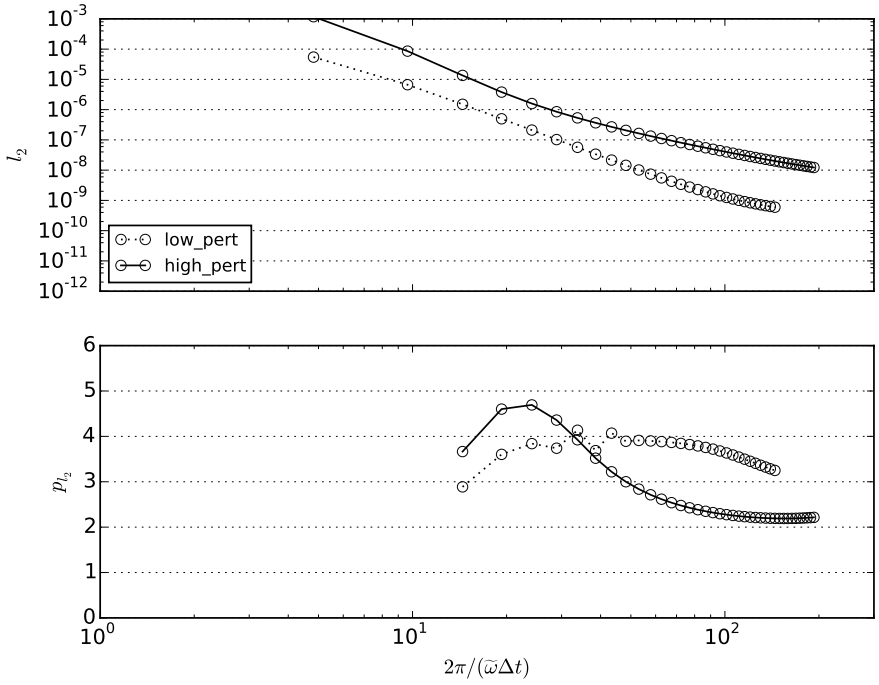

Figure 25. ρ l_2 error and convergence rate for BASS's RK4L scheme with low- and high-amplitude Gaussian perturbations. Equation (30) was used to calculate p_{l_2}.

$\widetilde{v} = 0.08$, $\widetilde{w} = 0.0075$, $\widetilde{p} = \frac{1}{10\gamma}$. The density l_2 results for this high-perturbation test case are compared to the original in Figure 25. As might be expected, the larger Gaussian amplitudes increase the l_2 error markedly, but also reduce the RK4L's convergence rate to 2, the expected value for non-linear differential equations.

The RK56's $l_{\max}$ error exhibits an interesting behavior for the ρw variable. Figure 26 compares this data with the corresponding l_2 case. The p_{l_2} curve is fairly close to the expected fourth-order convergence, but $p_{l_{\max}}$ is more like 3.5, a quality not seen in the other flow variables or time-marching schemes. The explanation may lie in the assumed form of the error used for calculating the convergence rate. For (30), the assumption

$$\epsilon_{i,j} \approx A\Delta x_i^p + B\Delta t_j^q \tag{48}$$

was made. An improved expression may be

$$\epsilon_{i,j} \approx A\Delta t\Delta x_i^p + B\Delta t_j^q. \tag{49}$$

If (49) holds, one would expect the second term to dominate the error for most of the time steps in Figure 26, since the grid spacing is quite small. As the time step size is reduced, however, the two terms in (49) may become closer in magnitude, which would tend to degrade the convergence rate calculation. This would especially be true for the RK56, the most accurate time-marching scheme, and the very sensitive $l_{\max}$ norm.

V. Conclusions and Future Work

In this work, the EVA approach to code verification was extended to verify the non-linear Navier-Stokes equations for the first time. Unlike the Method of Manufactured Solutions (MMS), the current state-of-the-art code verification technique, EVA does not require the addition of complicated source terms to a code's governing equations, and thus the code itself. A survey of the relevant PDE code literature appears to show that the MMS source terms limit the technique's popularity among researchers. Unlike the Method of Exact Solutions (MES), EVA maintains most of the flexibility in the form of the reference solution that MMS enjoys.

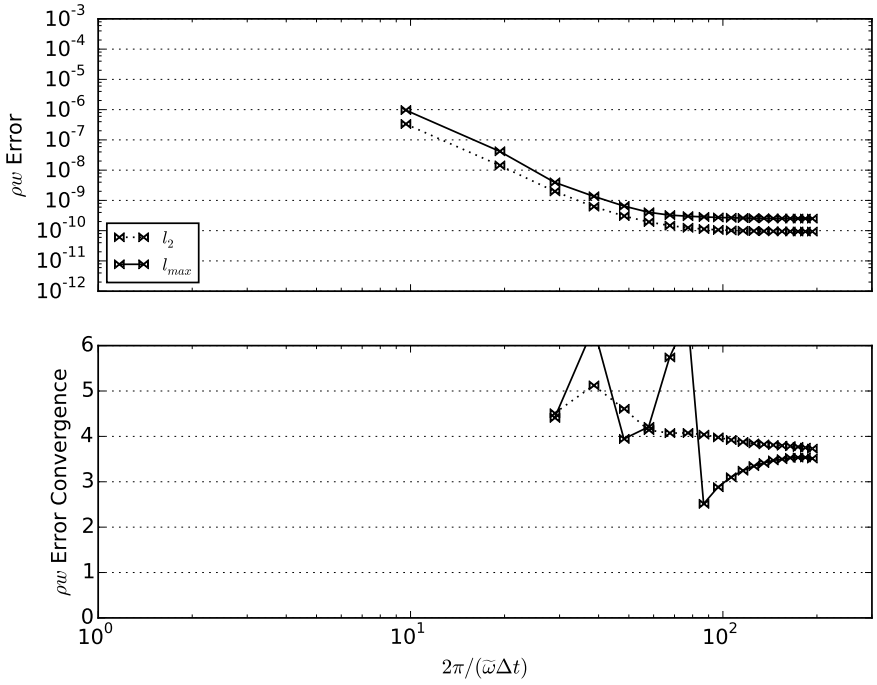

Figure 26. ρw l_2 **and** $l_{\max}$ **data for BASS's RK56 scheme. Equation** (30) **was used to calculate both convergence rates.**

The EVA solution technique was explained briefly through its application to the linear advection equation, and then in some detail with regards to the Navier-Stokes equations. Pseudocode of the current EVA implementation were included and explained. The recurrence relations needed to construct the EVA solution are found in the appendix of this work.

The EVA tool was then used to verify the viscous form of the NASA Glenn's Broadband Aeroacoustic Stator Simulator (BASS), a high-order, parallel Computational Aeroacoustics code. After a Fourier analysis of BASS's numerical schemes, verification results for the code's spatial and temporal schemes were presented. For the most part, the results closely mirrored what was predicted by the Fourier analysis. Happily, the error from each of BASS's schemes attained their design convergence rate, strongly indicating all are coded properly.

The most significant limitation of the EVA method is the inability to specify boundary conditions along domain surfaces. In principle, it should be possible to use one of the spatial variables as the Cauchy marching direction instead of time, which would allow one to enforce a boundary condition at the cost of relinquishing control over the initial flow. This approach would require that the surface is not characteristic, and would not be applicable to solid wall boundary conditions.

Another difficult aspect of code verification is determining the right set of grid spacing, time step size, and flow parameters that will allow the asymptotic range of the numerical scheme(s) under consideration to be found without extremely dense grids, many time steps, etc.. Assuming a more general form of the error than (28) for the order-of-accuracy calculation, and possibly refining grid spacing and time step size simultaneously, may be more flexible and robust.

Acknowledgments

This work was supported by the Subsonic Fixed Wing Project of the NASA Fundamental Aeronautics Program. The technical monitor was Dr. Edmane Envia of the NASA Glenn Research Center.

Appendix

The recurrence relation for the three-dimensional Navier-Stokes continuity equation used in this work is

$$\begin{aligned}
\frac{\partial^{b+d+f+n+1}\sigma}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^{n+1}} = &-\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
&\frac{\partial^{b-a+d-c+f-e+n-m}u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+1+c+e+m}\sigma}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{\partial^{b-a+d-c+f-e+n-m}v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+c+1+e+m}\sigma}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{\partial^{b-a+d-c+f-e+n-m}w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+c+e+1+m}\sigma}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}}\\
&-\frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\Bigg(\\
&\frac{\partial^{a+1+c+e+m}u}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{\partial^{a+c+1+e+m}v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{\partial^{a+c+e+1+m}w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}}\Bigg)\Bigg].
\end{aligned}\tag{16c repeated}$$

and the corresponding recurrence relation for the x-momentum equation is

$$\begin{aligned}
\frac{\partial^{b+d+f+n+1}u}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^{n+1}} = &-\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
&\frac{\partial^{b-a+d-c+f-e+n-m}u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+1+c+e+m}u}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{\partial^{b-a+d-c+f-e+n-m}v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+c+1+e+m}u}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{\partial^{b-a+d-c+f-e+n-m}w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+c+e+1+m}u}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}}\\
&+\frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\frac{\partial^{a+1+c+e+m}p}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}}\\
&-\mu\frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\Bigg(\frac{4}{3}\frac{\partial^{a+2+c+e+m}u}{\partial x^{a+2}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{1}{3}\frac{\partial^{a+1+c+1+e+m}v}{\partial x^{a+1}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}}\\
&+\frac{1}{3}\frac{\partial^{a+1+c+e+1+m}w}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}}\\
&+\frac{\partial^{a+c+2+m+m}u}{\partial x^{a}\,\partial y^{c+2}\,\partial z^{m}\,\partial t^{m}}+\frac{\partial^{a+c+e+2+m}u}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+2}\,\partial t^{m}}\Bigg)\\
&\Bigg],
\end{aligned}\tag{50}$$

and the corresponding recurrence relation for the y-momentum equation is

$$\begin{aligned}
\frac{\partial^{b+d+f+n+1} v}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^{n+1}} = -\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
&\frac{\partial^{b-a+d-c+f-e+n-m} u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m} v}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+n-m} v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m} v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+n-m} w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m} v}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+n-m} \sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m} p}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&-\mu\frac{\partial^{b-a+d-c+f-e+n-m} \sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\Bigg(\frac{4}{3}\frac{\partial^{a+c+2+e+m} v}{\partial x^{a}\,\partial y^{c+2}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{1}{3}\frac{\partial^{a+1+c+1+e+m} u}{\partial x^{a+1}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{1}{3}\frac{\partial^{a+c+1+e+1+m} w}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{\partial^{a+2+c+m+m} v}{\partial x^{a+2}\,\partial y^{c}\,\partial z^{m}\,\partial t^{m}}+\frac{\partial^{a+c+e+2+m} v}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+2}\,\partial t^{m}}\Bigg) \\
&\Bigg],
\end{aligned} \tag{51}$$

and, for the z-momentum equation,

$$\begin{aligned}
\frac{\partial^{b+d+f+n+1} w}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^{n+1}} = -\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
&\frac{\partial^{b-a+d-c+f-e+n-m} u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m} w}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+n-m} v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m} w}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+n-m} w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m} w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+n-m} \sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&-\mu\frac{\partial^{b-a+d-c+f-e+n-m} \sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\Bigg(\frac{4}{3}\frac{\partial^{a+c+e+2+m} w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+2}\,\partial t^{m}} \\
&+\frac{1}{3}\frac{\partial^{a+1+c+e+1+m} u}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{1}{3}\frac{\partial^{a+c+1+e+1+m} v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{\partial^{a+2+c+m+m} w}{\partial x^{a+2}\,\partial y^{c}\,\partial z^{m}\,\partial t^{m}}+\frac{\partial^{a+c+2+e+m} w}{\partial x^{a}\,\partial y^{c+2}\,\partial z^{e}\,\partial t^{m}}\Bigg) \\
&\Bigg].
\end{aligned} \tag{52}$$

Finally, the energy equation recurrence relation is

$$
\begin{aligned}
\frac{\partial^{b+d+f+n+1} p}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^{n+1}} = -\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
\frac{\partial^{b-a+d-c+f-e+n-m} u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m} p}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+f-e+n-m} v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m} p}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+f-e+n-m} w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
+\gamma\frac{\partial^{b-a+d-c+f-e+n-m} p}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\Bigg(\\
\frac{\partial^{a+1+c+e+m} u}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{a+c+1+e+m} v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{a+c+e+1+m} w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}}\Bigg) \\
-(\gamma-1)\left(\frac{\partial^{b+d+f+n}\Theta}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^n}+\frac{\partial^{b+d+f+n}\Phi}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^n}\right) \\
\Bigg],
\end{aligned}
\tag{53}
$$

where

$$
\begin{aligned}
\frac{\partial^{b+d+n+n}\Theta}{\partial x^b\,\partial y^d\,\partial z^n\,\partial t^n} = \frac{\mathrm{k}}{\mathrm{R}}\sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
\frac{\partial^{b-a+2+d-c+f-e+n-m}\sigma}{\partial x^{b-a+2}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+2\frac{\partial^{b-a+1+d-c+f-e+n-m}\sigma}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m} p}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+2+c+e+m} p}{\partial x^{a+2}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+2+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c+2}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+2\frac{\partial^{b-a+d-c+1+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c+1}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m} p}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+2+e+m} p}{\partial x^{a}\,\partial y^{c+2}\,\partial z^{e}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+f-e+2+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e+2}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
+2\frac{\partial^{b-a+d-c+f-e+1+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e+1}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
+\frac{\partial^{b-a+d-c+f-e+n-m}\sigma}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+2+m} p}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+2}\,\partial t^{m}} \\
\Bigg]
\end{aligned}
\tag{54}
$$

and

$$
\begin{aligned}
\frac{\partial^{b+d+f+n}\Phi}{\partial x^b\,\partial y^d\,\partial z^f\,\partial t^n} = \mu \sum_{a=0}^{b}\binom{b}{a}\sum_{c=0}^{d}\binom{d}{c}\sum_{e=0}^{f}\binom{f}{e}\sum_{m=0}^{n}\binom{n}{m}\Bigg[\\
&\frac{4}{3}\Bigg(\\
&\quad \frac{\partial^{b-a+1+d-c+f-e+n-m}u}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m}u}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+1+f-e+n-m}v}{\partial x^{b-a}\,\partial y^{d-c+1}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m}v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+1+n-m}w}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e+1}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m}w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&-\frac{\partial^{b-a+1+d-c+f-e+n-m}u}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m}v}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&-\frac{\partial^{b-a+1+d-c+f-e+n-m}u}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m}w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&-\frac{\partial^{b-a+d-c+1+f-e+n-m}v}{\partial x^{b-a}\,\partial y^{d-c+1}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m}w}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&\Bigg) \\
&+\frac{\partial^{b-a+1+d-c+f-e+n-m}v}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m}v}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+1+d-c+f-e+n-m}w}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m}w}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+1+f-e+n-m}u}{\partial x^{b-a}\,\partial y^{d-c+1}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m}u}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+1+f-e+n-m}w}{\partial x^{b-a}\,\partial y^{d-c+1}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m}w}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+1+n-m}u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e+1}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m}u}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+1+n-m}v}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e+1}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m}v}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&+2\Bigg(\\
&\quad \frac{\partial^{b-a+1+d-c+f-e+n-m}v}{\partial x^{b-a+1}\,\partial y^{d-c}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+1+e+m}u}{\partial x^{a}\,\partial y^{c+1}\,\partial z^{e}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+1+f-e+n-m}w}{\partial x^{b-a}\,\partial y^{d-c+1}\,\partial z^{f-e}\,\partial t^{n-m}}\,\frac{\partial^{a+c+e+1+m}v}{\partial x^{a}\,\partial y^{c}\,\partial z^{e+1}\,\partial t^{m}} \\
&+\frac{\partial^{b-a+d-c+f-e+1+n-m}u}{\partial x^{b-a}\,\partial y^{d-c}\,\partial z^{f-e+1}\,\partial t^{n-m}}\,\frac{\partial^{a+1+c+e+m}w}{\partial x^{a+1}\,\partial y^{c}\,\partial z^{e}\,\partial t^{m}} \\
&\Bigg) \\
\Bigg].
\end{aligned}
\tag{55}
$$

References

[1]Oberkampf, W. L. and Roy, C. J., *Verification and Validation in Scientific Computing*, Cambridge University Press, Oct. 2010.

[2]Blottner, F. G., "Accurate Navier-Stokes results for the hypersonic flow over a spherical nosetip," *Journal of Spacecraft and Rockets*, Vol. 27, No. 2, 1990, pp. 113–122.

[3]Roache, P. J., *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Jan. 1998.

[4]Roache, P. J., "Building PDE Codes to be Verifiable and Validatable," *Computing in Science and Engineering*, Vol. 6, No. 5, 2004, pp. 30–38.

[5]Knupp, P. and Salari, K., *Verification of Computer Codes in Computational Science and Engineering*, CRC Press, Oct. 2002.

[6]Steinberg, S. and Roache, P. J., "Symbolic manipulation and computational fluid dynamics," *Journal of Computational Physics*, Vol. 57, No. 2, Jan. 1985, pp. 251–284, 00079.

[7]Roache, P. J., "Verification of codes and calculations," *AIAA journal*, Vol. 36, No. 5, 1998, pp. 696–702.

[8]Salari, K. and Knupp, P., "Code Verification by the Method of Manufactured Solutions," Tech. Rep. SAND2000-1444, Sandia National Laboratories, June 2000.

[9]Roache, P. J., "Code verification by the method of manufactured solutions," *Journal of Fluids Engineering*, Vol. 124, No. 1, 2002, pp. 4–10.

[10]Veluri, S. P., Roy, C. J., Hebert, S., and Luke, E. A., "Verification of the Loci-CHEM CFD Code Using the Method of Manufactured Solutions," *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[11]Veluri, S. P., *Code verification and numerical accuracy assessment for finite volume CFD codes*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2010.

[12]Veluri, S., Roy, C. J., and Luke, E., "Comprehensive code verification for an unstructured finite volume CFD code," *48th AIAA Aerospace Sciences Meeting*, Vol. 127, 2010.

[13]Veluri, S. P., Roy, C. J., and Luke, E. A., "Comprehensive code verification techniques for finite volume {CFD} codes," *Computers & Fluids*, Vol. 70, No. 0, 2012, pp. 59 – 72.

[14]Tremblay, D., Etienne, S., and Pelletier, D., "Code Verification and the Method of Manufactured Solutions for Fluid-Structure Interaction Problems," *36th AIAA Fluid Dynamics Conference and Exhibit*, American Institute of Aeronautics and Astronautics.

[15]Étienne, S., Garon, A., and Pelletier, D., "Some manufactured solutions for verification of fluid-structure interaction codes," *Computers & Structures*, Vol. 106–107, No. 0, 2012, pp. 56 – 67.

[16]Gong, Z., Lu, C., and Huang, H., "Accuracy analysis of immersed boundary method using method of manufactured solutions," *Applied Mathematics and Mechanics*, Vol. 31, No. 10, Oct. 2010, pp. 1197–1208.

[17]Eca, L. and Hoekstra, M., "An introduction to CFD code verification including eddy-viscosity Models," *ECCOMAS CFD*, Vol. 2006, 2006.

[18]Eça, L., Hoekstra, M., Hay, A., and Pelletier, D., "On the construction of manufactured solutions for one and two-equation eddy-viscosity models," *International Journal for Numerical Methods in Fluids*, Vol. 54, No. 2, 2007, pp. 119–154.

[19]Merroun, O., Almers, A., Bardouni, T. E., Bakkari, B. E., and Chakir, E., "Analytical benchmarks for verification of thermal-hydraulic codes based on sub-channel approach," *Nuclear Engineering and Design*, Vol. 239, No. 4, 2009, pp. 735 – 748.

[20]Pautz, S. D., "Verification of transport codes by the method of manufactured solutions: the ATTILA experience," *ANS International Meeting on Mathematical Methods for Nuclear Applications, Salt Lake City, Utah, American Nuclear Society*, 2001.

[21]Oberkampf, W. L., Trucano, T. G., and Pilch, M. M., "On the Role of Code Comparisons in Verification and Validation." Tech. Rep. SAND2003-2752, Sandia National Laboratories, Aug. 2003.

[22]Ingraham, D. and Hixon, R., "External verification analysis: A code-independent verification technique for unsteady PDE codes," *Journal of Computational Physics*, Vol. 243, June 2013, pp. 46–57.

[23]Garabedian, P. R., *Partial Differential Equations*, American Mathematical Society, 1964.

[24]Folland, G. B., *Introduction to Partial Differential Equations*, Princeton University Press, 1976.

[25]"The Python Programming Language," `https://www.python.org/`.

[26]SymPy Development Team, "SymPy: Python library for symbolic mathematics," `http://www.sympy.org`, 2014.

[27]Hixon, R., Nallasamy, M., and Sawyer, S., "Parallelization Strategy for an Explicit Computational Aeroacoustics Code," *8th AIAA/CEAS Aeroacoustics Conference & Exhibit*, American Institute of Aeronautics and Astronautics, Breckenridge, Colorado, June 2002.

[28]Tam, C. K. W. and Webb, J. C., "Dispersion-Relation-Preserving Finite Difference Schemes for Computational Acoustics," *Journal of Computational Physics*, Vol. 107, No. 2, Aug. 1993, pp. 262–281.

[29]Tam, C. and Shen, H., "Direct computation of nonlinear acoustic pulses using high-order finite difference schemes," *15th Aeroacoustics Conference*, American Institute of Aeronautics and Astronautics, 1993.

[30]Hixon, R., "Prefactored Small-Stencil Compact Schemes," *Journal of Computational Physics*, Vol. 165, No. 2, Dec. 2000, pp. 522–541, 00079.

[31]Lele, S. K., "Compact finite difference schemes with spectral-like resolution," *Journal of Computational Physics*, Vol. 103, No. 1, Nov. 1992, pp. 16–42.

[32]Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," *AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, HI, June 1991, AIAA 91-1596.

[33]Stanescu, D. and Habashi, W. G., "2N-Storage Low Dissipation and Dispersion Runge-Kutta Schemes for Computational Acoustics," *Journal of Computational Physics*, Vol. 143, No. 2, 1998, pp. 674–681.

[34]Allampalli, V., Hixon, R., Nallasamy, M., and Sawyer, S. D., "High-accuracy large-step explicit Runge–Kutta (HALE-RK) schemes for computational aeroacoustics," *Journal of Computational Physics*, Vol. 228, No. 10, June 2009, pp. 3837–3850.

[35]Giles, M. B., "Nonreflecting boundary conditions for Euler equation calculations," *AIAA Journal*, Vol. 28, No. 12, 1990, pp. 2050–2058, 00601.

[36]Butcher, J. C., *Numerical methods for ordinary differential equations*, Wiley, Chichester, England ; Hoboken, NJ, 2nd ed., 2008.